



José Ricardo de Lima Abrantes

Licenciatura em Ciências de Engenharia Electrotécnica e de Computadores

Ponto de Acesso Móvel em Ambiente Sensorial

Dissertação para obtenção do Grau de Mestre em
Engenharia Electrotécnica e de Computadores

Orientador: Tiago Oliveira Cardoso, Doutor, FCT



FACULDADE DE
CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA

Setembro 2015

Ponto de Acesso Móvel em Ambiente Sensorial

Copyright © José Ricardo de Lima Abrantes (31801), Faculdade de Ciências e Tecnologia, Universidade Nova de Lisboa

A Faculdade de Ciências e Tecnologia e a Universidade Nova de Lisboa têm o direito, perpétuo e sem limites geográficos, de arquivar e publicar esta dissertação através de exemplares impressos reproduzidos em papel ou de forma digital, ou por qualquer outro meio conhecido ou que venha a ser inventado, e de a divulgar através de repositórios científicos e de admitir a sua cópia e distribuição com objectivos educacionais ou de investigação, não comerciais, desde que seja dado crédito ao autor e editor.

Agradecimentos

Agradeço ao Sr. Professor Orientador Tiago Cardoso por me ter dado a oportunidade de poder fazer este projecto, pela sua ajuda, pela pressão que me colocou para continuar e conseguir entregar esta dissertação no suposto tempo.

Agradeço ao Sr. Eng. Rui Henriques, por me ter dado ideias sobre o que fazer e como fazer, sempre chamando a atenção a pormenores, o que sem dúvida ajudou na conclusão deste projecto. Agradeço também por me ter dado a conhecer um pouco do ambiente da empresa que nos acolheu, durante a realização desta dissertação.

Aos meus pais, por me terem dado a oportunidade de poder estudar e tirar um curso de que gosto. Obrigado também por terem paciência, quando o trabalho apertava demais e acabava por responder mal. Obrigado por me darem tudo o que precisei e estarem presentes nos momentos mais importantes da minha vida e quando mais precisei. Obrigado pela compreensão, quando se ia almoçar e saía antes, para ir estudar.

À minha namorada Marisa, por me ter ajudado em momentos difíceis. Pela compreensão, devido a por vezes não poder estar com ela, porque tinha de estudar. Pelo conforto que me deu quando mais precisei e por me ajudar nesta altura da minha vida.

Ao meu amigo David Henriques, pelas lutas que fomos travando ao longo destes anos, seja trabalhos que fizemos em conjunto, como noutras situações não relacionadas com a faculdade. Espero que a batalha continue daqui para a frente, mas se não for pedir muito, com mais horas de sono.

Agradeço também ao grupo de amigos, SAL, que sem dúvida me ajudou a desanuviar quando o trabalho parecia não acabar.

Aos meus amigos em geral que de uma forma ou de outra, me acompanharam neste trajecto e me ajudaram a ser o que sou hoje.

Sumário

Actualmente a maioria das pessoas pretendem ter controlo sobre tudo o que lhes pertence, quer seja saber a temperatura a que está uma sala, saber se um bebé está a chorar, detectar a presença de outras pessoas, entre outras utilidades que possam ser extraídas recorrendo a diferentes sensores. Para além de quererem essa informação, desejam-na o mais rápido possível. De forma a ter-se acesso à mesma, é necessário que estes equipamentos tenham sensores acoplados. Em casos que envolvam mais do que um sensor, a solução actual passa por ter uma rede ligada a um *gateway*/ ponto de acesso. Este ponto de acesso necessita ter uma ligação à internet para poder enviar os dados de forma a serem visualizados por um cliente. Esta ligação pode ser feita por cabo, o que por vezes é impossível, tendo em conta a localização do equipamento. Ou poderá ser necessário recorrer a um cartão 3G/ 4G, sendo para isso, obrigado a pagar uma mensalidade.

Tendo em conta este factor, uma possível solução, passa por utilizar um equipamento que nos acompanha no nosso quotidiano, como por exemplo um telemóvel, permitindo assim criar um sistema onde o *gateway* é o telemóvel, sendo que a informação proveniente da rede de sensores seria enviada através deste.

Nesta dissertação apresenta-se uma solução composta de elementos responsáveis por: recolha de informação sensorial, processamento da informação e disponibilização da mesma via Web.

O telemóvel terá a capacidade de armazenar vários tipos de informação, sendo que, assim que possível, enviará a mesma para uma pilha de dados presente no servidor. Existirá uma aplicação que irá consumir essa pilha, de forma a tratar os dados, para posteriormente os armazenar num local adequado. Neste servidor estará ainda uma página alojada, permitindo assim, efectuar o *login* ou o registo.

No servidor de um suposto cliente existirá também uma página, onde serão mostrados os dados que foram recolhidos.

Palavras-chave: base de dados, Bluetooth, gateway, smartphone, web service, sensores, Android

Abstract

Nowadays, the majority of people intend to obtain control about everything that belongs to them, like knowing the room temperature, knowing if the baby is crying, detecting the presence of other persons, among other utilities that can be taken out making use of different sensors. In spite of wanting that information, they wish it as fast as possible.

In cases in which more than one sensor is involved, the actual solution goes by having a network connected to a gateway/ access point. This access point needs to have an internet connection to be able to send the data in order to be viewed by a client. This connection can be made by cable, which sometimes is impossible taking into account the position of the equipment. Or it can be used a 3G/ 4G card, but for that it is necessary to pay a monthly fee.

Taking this factor into consideration, a possible solution goes by using an equipment which accompanies us in our everyday life, like a mobile phone, for example, allowing in this way to create a system where the gateway is the mobile phone, thus the information from the sensor network would be sent through this.

In this dissertation it is presented a solution composed by responsible elements for: gathering of sensory information, information processing and availability of the same via web.

The mobile phone will have the capacity to store several types of information, then, as soon as possible, these informations will be sent to a stack of data present in the server. There will be an application which will consume that stack so that it can treat them in order to store them later on in an adequate place. Still in this server it will be a hosted page allowing to make login or registry.

In the server of a supposed client there will be also a page, where the data which was gathered will be shown.

Keywords: database, Bluetooth, gateway, smartphone, web service, sensor, Android

Acrónimos

API - Application Programming Interface
AMPS - Advanced Mobile Phone System
AES - Advanced Encryption Standard
CDMA - Code Division Multiple Access
D-AMPS – Digital AMPS
Daas – Database as a service
DBMS - Database Management System
DNS – Domain Name Service
EAP - Extensible Authentication Protocol
EDGE - Enhanced Data rates for GSM Evolution
FDMA - Frequency Division Multiple Access
GSM – Global System for Mobile Communications
GPRS - General Packet Radio Service
HSCSD - High-Speed Circuit-Switched Data
HSPA - High Speed Packet Access
HSDPA - High Speed Downlink Packet Access
HSUPA - High Speed Uplink Packet Access
IEEE - Institute of Electrical and Electronics Engineers
IaaS – Infrastructure as a service
IDS - Intrusion Detection System
IOS – Iphone OS
IP – Internet Protocol
IPvx – IP version x
ISP – Internet Service Provider
Json - JavaScript Object Notation
Km – Quilómetro
LAN – Local Area Network
LTE – Long Term Evolution
MAN – Metropolitan Area Network

MAC – Media Access Control
MD5 – Message Digest 5
MIMO – Multiple Input Multiple Output
NFC – Near Field Communication
NOSQL – Not Only SQL
OFDMA – Orthogonal FDMA
Paas – Plataform as a service
PSK – Pre-shared Key
RSA – Rivest, Shamir, Adleman
Saas – Software as a service
SSID – Service Set ID
SSL - Secure Socket Layer
SQL - Structured Query Language
SHA – Secure Hash Algoritm
SIM - Subscriber Identity Module
TLS - Transport Layer Security
TKIP - Temporal Key Integrity Protocol
Taas – Testing as a service
UMTS - Universal Mobile Telecommunications System
WAN – Wide Area Network
WPAN – Wireless Personal Area Network
WLAN – Wireless LAN
WMAN – Wireless MAN
WWAN – Wireless WAN
WEP - Wired Equivalent Privacy
WAP – Wi-Fi Protected Access
Wi-Fi – Wireless Fidelity
W-CDMA – Wide CDMA

Índice

1	Introdução	- 1 -
1.1	Enquadramento e motivação	- 1 -
1.2	Problema	- 3 -
1.3	Objectivo	- 3 -
1.4	Organização da dissertação	- 5 -
2	Estado da arte	- 7 -
2.1	Plataformas móveis e serviços <i>cloud</i>	- 7 -
2.2	Sensores	- 9 -
2.3	Redes em malha	- 10 -
2.4	Segurança	- 11 -
2.5	Projectos semelhantes	- 19 -
2.6	Tecnologias Sem fios <i>versus</i> com fios	- 22 -
2.6.1	Tipos de comunicação sem fios.....	- 24 -
2.6.2	Vantagens/desvantagens	- 27 -
2.7	<i>Cloud</i> computing	- 28 -
2.7.1	O que é?	- 28 -
2.7.2	Vantagens/desvantagens	- 28 -
2.7.3	Modelo computacional <i>Cloud</i>	- 29 -
2.7.4	Exemplos	- 32 -
2.8	Evolução dos telefones móveis/das redes móveis.....	- 32 -
2.9	Sistemas Operativos	- 39 -
2.9.1	Android.....	- 39 -
2.9.2	IOS	- 40 -
2.10	Evolução da internet IPv4/IPv6	- 41 -
2.11	<i>Web services</i>	- 46 -
2.12	Base de dados	- 47 -
3	Proposta	- 51 -
3.1	Proposta de solução	- 51 -
3.2	Arquitectura da solução proposta.....	- 53 -
4	Validação	- 61 -
4.1	Passos de implementação	- 61 -
4.2	Análise das tecnologias	- 62 -

4.3	Resultados de simulação	- 65 -
4.4	Teste de performance	- 81 -
5	Conclusão e Trabalhos Futuros	- 83 -
6	Bibliografia.....	- 87 -
7	Anexo.....	- 93 -

Índice de figuras

Figura 2.1 - Exemplo de uma <i>cloud</i> , em específico a Dropbox [4]	- 8 -
Figura 2.2 - Exemplo genérico de uma <i>cloud</i> [5].....	- 8 -
Figura 2.3 - NTX com quatro sensores ligados aos inputs [6]	- 9 -
Figura 2.4 - Diferentes tipologias [10].....	- 11 -
Figura 2.5 – Descrição de um endereço web [12].....	- 12 -
Figura 2.6 - Esquema de interacção com o SSL [13].....	- 13 -
Figura 2.7 - Configuração de um encaminhador.....	- 17 -
Figura 2.8 - Representação do esquema protótipo da Link [21].....	- 19 -
Figura 2.9 - Exemplo de configuração do Valarm [23]	- 20 -
Figura 2.10 - Arquitectura do produto da empresa Libelium [25]	- 21 -
Figura 2.11 - SensorCloud [26]	- 21 -
Figura 2.12 - Imagem demonstrativa dos três tipos de rede [27] [28]	- 22 -
Figura 2.13 - Exemplo de indicação de um encaminhador	- 25 -
Figura 2.14 - O mundo da <i>cloud</i> [37].....	- 28 -
Figura 2.15 - Comparação de velocidade <i>versus</i> distância com várias tecnologias [50].....	- 36 -
Figura 2.16 - Indicador de redes móveis [51].....	- 36 -
Figura 2.17 - Indicações numa caixa de um <i>smartphone</i>	- 37 -
Figura 2.18 - Exemplo de um <i>cluster</i> constituído por 7 células [44]	- 38 -
Figura 2.19 - Arquitectura do sistema Android [53]	- 39 -
Figura 2.20 - Camadas do sistema operativo IOS.....	- 41 -
Figura 2.21 - Endereço IP6 compatível com IPv4 [64].....	- 44 -
Figura 2.22 - À esquerda <i>ping</i> com IPv4 e à direita <i>ping</i> com IPv6.....	- 44 -
Figura 2.23 - Transmissão de pacotes através de túnel IPv6 em IPv4 [64]	- 45 -
Figura 2.24 - Estrutura de uma mensagem SOAP [66].....	- 46 -
Figura 2.25 - Modelo cliente-servidor [67]	- 47 -
Figura 2.26 -Imagem demonstrativa de SQL e NOSQL [72]	- 49 -
Figura 3.1 - Proposta de solução 1, presente na empresa.....	- 51 -
Figura 3.2 - Proposta de solução 2, presente na empresa.....	- 52 -
Figura 3.3 - Proposta de solução 3, presente na empresa.....	- 52 -
Figura 3.4 - Proposta de solução 4, presente na empresa e no cliente	- 52 -
Figura 3.5 -Arquitectura do projecto	- 53 -
Figura 3.6 - Arquitectura do projecto em blocos	- 54 -

Figura 3.7 -Consumidor a ser executado no putty.....	- 56 -
Figura 3.8 - À esquerda um consumidor a correr no Eclipse e à direita o emulador do Android com a aplicação.....	- 56 -
Figura 3.9 -Imagem do smartphone.....	- 57 -
Figura 3.10 - À esquerda emulador do Android e à direita consumidor a correr no eclipse.....	- 57 -
Figura 3.11 - Nome de canal do Broker.....	- 58 -
Figura 3.12 - Ligação da aplicação em java ao Broker.....	- 58 -
Figura 4.1 - Montagem do Arduíno com Bluetooth e cartão SD.....	- 65 -
Figura 4.2 – Referente às duas aplicações que foram usadas para testes.....	- 65 -
Figura 4.3 - O HMSoft a receber informação do cartão simulando as medições dos sensores em formato Json.....	- 66 -
Figura 4.4 - Vários estados de login	- 66 -
Figura 4.5 - Função para guardar dados na CouchDBLite no smartphone.....	- 67 -
Figura 4.6 - Programação implementada em Android para lidar com o pedido para guardar dados na CouchDBLite	- 67 -
Figura 4.7 - Esquema de <i>login</i> e envio de mensagens	- 67 -
Figura 4.8 - Programação na aplicação em java que lida com a recepção de mensagens	- 68 -
Figura 4.9 - Páginas de <i>login</i> , particular e organização.....	- 68 -
Figura 4.10 - Programação na aplicação em java que lida com um pedido de <i>login</i>	- 69 -
Figura 4.11 - Página inicial dos utilizadores e das organizações	- 70 -
Figura 4.12 - Exemplo de adicionar um utilizador a uma organização	- 70 -
Figura 4.13 - Página de registo de particulares e organizações	- 70 -
Figura 4.14 - Serviço do <i>Web Service</i> register_new_user	- 71 -
Figura 4.15 - Página de login de um suposto cliente.....	- 72 -
Figura 4.16 - Página inicial do cliente.....	- 72 -
Figura 4.17 - Página com informação de sensores.....	- 73 -
Figura 4.18 - Adicionar sensor a uma organização.....	- 73 -
Figura 4.19 - Mensagem de confirmação.....	- 73 -
Figura 4.20 - Página com informação de sensores actualizado	- 74 -
Figura 4.21 - Exemplo usado no projecto que demonstra a forma como se cria um cliente SOAP ...	- 74 -
Figura 4.22 - Exemplo de chamar uma função com parâmetros	- 75 -
Figura 4.23 - <i>Web service</i> a correr no GlassFish.....	- 75 -
Figura 4.24 - Serviços criados no netbeans que estão disponíveis no <i>web service</i>	- 75 -

Figura 4.25 -Serviço disponibilizado no <i>web service</i> que é usado para se conectar à base de dados -	
76 -	
Figura 4.26 -Função GetLogin do <i>web service</i>	- 77 -
Figura 4.27 -Interface Futon do CouchDB na parte das <i>view</i> e respectivo código	- 78 -
Figura 4.28 - Ficheiro exemplo do que é guardado no CouchDB.....	- 79 -
Figura 4.29 - Resposta do CouchDB a uma <i>view</i>	- 79 -
Figura 4.30 - Teste de performance feito à base de dados couchDB	- 81 -

Índice de Tabelas

Tabela 2.1- Tabela demonstrativa de evolução do <i>share</i> do mercado. [1].....	- 7 -
Tabela 2.2 – Comparação entre várias tecnologias [36]	- 27 -
Tabela 2.3 - Tabela comparativa de <i>ping</i> entre tecnologia 3G e 4G [46].....	- 35 -

1 Introdução

1.1 Enquadramento e motivação

Nos dias de hoje, as pessoas querem ter tudo ao seu alcance, desejam ter a informação sempre acessível e interligada. Quer se tratem de *smartphones*, carros ou casas inteligentes, electrodomésticos, *gadgets*, basicamente algo que possa ser ligado à internet. As pessoas desejam ter controlo sobre os mesmos, ver informação em tempo real, em qualquer lugar. De forma a ter acesso à mesma, é imprescindível que estes equipamentos tenham sensores acoplados. Para que seja posteriormente visualizada, é necessário que essa informação seja passada através de algum mecanismo de comunicação até alcançar o cliente. Tendo em conta que os dispositivos podem estar espalhados, e não terem uma ligação directa disponível a um ponto com internet, é necessário que estes dados sejam recolhidos de alguma forma. Uma destas formas seria enviar os dados através de um cartão 3G/ 4G, associado a um operador de redes móveis, onde é paga uma taxa mensal. É possível também pagar a alguém que se desloque até ao local, de forma a recolher a informação ou cablar até ao ponto de internet mais próximo, o que levaria a um custo elevado.

Esta dissertação foi desenvolvida em parceria com a empresa IrRADIARE. A IrRADIARE foca-se, essencialmente, nos sectores de energia. Está também presente no desenvolvimento de sistemas inovadores para responder a diversos problemas, entre os quais: gestão urbana, redução de custos de energia, optimização de processos de segurança e protecção do ambiente. Possui ainda uma visão sobre a sustentabilidade, desenvolvimento, competitividade e qualidade de vida.

Para obter estas informações é necessário recorrer a sensores. Dependendo do tipo de sensor, serão recolhidos diferentes tipos de informação. Por exemplo, para efectuar uma redução de custo na conta da electricidade, é possível instalar sensores que detectem a presença de pessoas, permitindo assim, que as luzes apenas se liguem quando efectivamente estiver alguém na divisão em questão. Existindo sensores que leiam a temperatura de uma sala ou verifiquem a humidade ou os níveis de poluição no ar, está-se a contribuir para uma melhoria na qualidade de vida. O facto de recorrer-se ao uso de sensores, permite que estes sejam um mecanismo de controlo para a activação do ar condicionado, desumidificador e extractores de ar, respectivamente.

Este projecto está dividido em duas partes. Uma primeira parte, de mais baixo nível, onde estão incluídos os sensores, uma organização em rede dos mesmos e a comunicação com um *gateway*, (parte desenvolvida na dissertação com o tema Sensores “em movimento” pelo colega David Filipe Serra Henriques). A segunda parte é desenvolvida nesta dissertação e aborda a componente de mais alto nível, que inclui a aplicação para o *gateway*, local onde se vai armazenar, gerir e aceder externamente à informação, através de funções e criação de site para disponibilização da mesma.

O projecto onde esta dissertação se insere tem como base os sensores. Os sensores estão ligados por cabos, a dispositivos que recebem a informação em intervalos de tempo regulares. A informação vai de dispositivo em dispositivo, dentro de uma rede em malha, até chegar ao dispositivo central. Esse dispositivo central é responsável por agrupar as mensagens dos diferentes dispositivos e enviá-las para um equipamento que está ligado directamente por cabo, que tem a capacidade de tratar informação (microprocessador).

Neste ponto tirar-se-á partido das pessoas que possuem um telemóvel, que passa dentro do alcance de quem está a emitir as mensagens, com o microprocessador acoplado. Caso, o telemóvel em questão, tenha uma ligação activa para poder receber mensagens, com a aplicação a correr, o microprocessador envia um pedido de conexão. Se a ligação for estabelecida, o telemóvel envia uma mensagem com a data mais recente, que possui informação na base de dados do telemóvel, relativamente ao sensor do qual está a ser enviado o pedido. Em seguida, o microprocessador procura todos os dados existentes no cartão, anteriores a essa data, e procede à sua eliminação, pois já se tem confirmação que esses dados se encontram na base de dados central. Após a eliminação dos dados anteriores a essa data, e que já se encontram na base de dados central, procede-se ao envio daqueles que ainda não se encontram armazenados, envio esse referente a uma hora de informação. Terminado o envio dos dados, a conexão com o telemóvel é concluída. Caso o telemóvel não tenha no momento uma ligação de internet, a aplicação será configurada, de modo a guardar essa informação numa base de dados interna. Quando o telemóvel tiver uma ligação à internet disponível, seja ela por Wi-Fi ou redes móveis, enviará os dados que dispõe para a aplicação que está a correr no servidor. Este envio recorre ao uso de um programa que permite criar uma pilha de dados no servidor, que serão consumidos pela aplicação. A aplicação que está a correr no servidor, vai estar conectada com uma base de dados e fazer todo o tipo de verificações necessárias, para não existirem dados repetidos nem guardados de forma errada, pois vão chegar mensagens repetidas à aplicação para se conseguir assegurar que os dados dos sensores são efectivamente guardados.

1.2 Problema

Esta dissertação foi realizada tendo em conta o problema que existia em enviar informação para a internet, quando não existia um *gateway*/ ponto de acesso disponível. Por exemplo, imaginando que existe uma rede de sensores que comunicam entre si, até chegar a um coordenador que receberá informação de todos os seus nós, e de sensores também. Esse coordenador está encarregue de enviar essa informação para um *gateway*/ ponto de acesso, necessitando assim de enviar dados para a *cloud*, através de uma ligação Wi-Fi, cabo ou cartão de dados. A ligação proveniente de um cartão de dados acarreta custos, pois todos os meses é necessário pagar uma mensalidade. Se for através de uma ligação sem fios, isso implica que exista um encaminhador por perto a emitir sinal, o que por vezes é impossível. Ou pode ser uma rede protegida e portanto não conseguirá acesso, impossibilitando assim o envio dos dados. Caso seja ligação por cabo, ainda se torna mais complicado, pois imaginando que o coordenador está no centro de um parque florestal a medir a qualidade do ar, será praticamente impossível que a ligação seja feita desta forma.

Existe também uma necessidade de confirmação que os dados enviados pelo ponto de acesso chegam realmente à internet, pois a intenção é conseguir recolher toda a informação.

Com a solução proposta ter-se-á a possibilidade de instalar uma rede de sensores, e poder visualizar os dados que estes recolhem, sem ter de recorrer a mudanças físicas no edifício, ou outro enquadramento.

1.3 Objectivo

Para esta dissertação, foi proposta a definição de uma estrutura base, através da qual um telemóvel passaria a ser um *gateway*, para permitir a ligação de um conjunto de sensores a uma rede central, de forma a essa informação poder ser armazenada, analisada e disponibilizada.

Para isto, será necessário uma aplicação que o telemóvel possa correr em segundo plano, devidamente autorizada e que permita enviar dados de sensores para a plataforma central. Deverá ter também a capacidade de conseguir gerir um vasto tipo de informação de sensores, desde dados biométricos, geográficos, ambientais, energéticos, etc. Também se poderá permitir que o *gateway* de um utilizador do serviço, possa recolher informação de sensores que não lhe pertencem, pois a instalação poderá ser de cariz pessoal.

O objectivo desta dissertação passa também por arranjar uma forma de comunicação, que seja mais económica que as actuais e que seja de fácil implementação. Não se pretende um *gateway*/ponto de ligação com a necessidade de uma ligação à internet, através de cabo que por vezes não está disponível, ou mesmo pagar a mensalidade que estaria associada ao uso de um cartão 3G, tenciona-se que este seja móvel. Para isso, é necessário que o *smartphone* que a pessoa utiliza habitualmente faça o trabalho de *gateway*. O *smartphone* possuirá a capacidade de guardar a informação, caso não tenha acesso à internet para enviar os dados naquele instante, evitando assim que os dados se percam. Também será necessário saber quais os últimos dados de cada sensor, que se encontram na base de dados. Irá ser também configurado um servidor com a aplicação que será desenvolvida neste projecto, que estará sempre à espera de uma ligação por parte de um cliente. Esse servidor irá conter a aplicação sempre em funcionamento e é essa mesma que vai tratar os dados. Existirá também um site, onde se poderá ver a informação que foi enviada.

Nesta dissertação apresentar-se-á uma solução que é diferente do que se encontra no mercado actual. Hoje em dia, existem muitos aparelhos que necessitam de estar ligados à internet, pois as pessoas precisam saber de forma rápida o que se passa em determinado local. Isto é possível com a recolha de dados de sensores, por exemplo. Estes dados para serem enviados requerem a passagem por um *gateway*/algum equipamento que consiga interligar a rede que recolhe a informação dos sensores e a rede de internet. No mercado actual é necessário recorrer a um ponto de ligação à internet que esteja, por exemplo, numa habitação, tratando-se de uma ligação fixa para se poder ter acesso por cabo. No caso de se tratar de um local remoto, esta hipótese não se coloca, pois normalmente não existe serviço de internet para que seja possível o envio de dados. Neste caso, a solução passa por comprar um *gateway* que tenha suporte a um cartão 3G. A solução proposta passa por fazer do *smartphone* que se usa regularmente, um *gateway*. Portanto, quando um *smartphone* estiver próximo do local de envio, recolheria a informação que este tinha armazenada e assim que tivesse oportunidade, e/ou, acesso à internet enviaria a informação e após isto a mesma estaria disponível para consulta. Com este método, poupar-se-ia numa possível mensalidade e por outro lado ganhar-se-ia em liberdade, pois é possível colocar os sensores onde se pretender, sem que para isso seja necessário ter cobertura de internet. Sendo assim uma enorme vantagem, pois poder-se-ia colocar este tipo de instalação num centro histórico, ou noutro local onde não se possa fazer mudanças físicas para o acesso à internet.

1.4 Organização da dissertação

A dissertação está dividida em 7 capítulos, os quais:

- Introdução – Introdução ao projecto realizado, explicando qual o problema e o objectivo.
- Estado da arte – É dado a conhecer o fundamento teórico, que sustenta o funcionamento que está implícito nas tecnologias, que estavam disponíveis para escolha. Assim como também foram enunciados os projectos semelhantes e as suas desvantagens/vantagens. Tal como o ponto de ligação entre as duas dissertações que constituem este projecto
- Proposta - Apresenta-se os módulos que constituem o projecto, assim como é feita a comunicação entre eles. Mostra-se também como é possível expandir a arquitectura
- Validação – Menciona os passos que foram tomados para chegar a uma proposta de solução. Resumo das tecnologias escolhidas ao longo da dissertação. Apresentação de resultados práticos e teste de performance.
- Conclusão e Trabalhos Futuros – São mencionados os vários problemas que foram encontrados ao longo da dissertação e os aspectos positivos de a ter efectuado. Enumera-se ainda os aspectos que são necessários melhorar para o projecto poder ser continuado
- Bibliografia – Referências bibliográficas
- Anexos – Aspectos relevantes a apresentar na dissertação

2 Estado da arte

Este capítulo descreve o actual estado da arte das áreas relacionadas com esta dissertação. Seja plataformas móveis, sensores, redes em malha, segurança e projectos semelhantes.

2.1 Plataformas móveis e serviços *cloud*

Actualmente, existem muitas plataformas disponíveis para dispositivos móveis, por exemplo: Android, IOS (Iphone Operating System), Windows phone, Blackberry, Firefox OS, Ubuntu touch OS. Existem muitos mais, mas que são baseados em Android, por exemplo, CyanogenMod, Fire OS, MIUI ou Flyme OS. A elevada percentagem de dispositivos Android, tal como se verifica na tabela 2.1, está também relacionada com o facto de existirem outros sistemas operativos baseados neste. Outros já foram descontinuados, mas, no entanto, são muito conhecidos por exemplo, o Symbian, Windows mobile, Palm os, WebOS, MeeGo e LiMo.

Período	Android	iOS	Windows Phone	BlackBerry OS	Outros
Q1 2015	78.0%	18.3%	2.7%	0.3%	0.7%
Q1 2014	81.2%	15.2%	2.5%	0.5%	0.7%
Q1 2013	75.5%	16.9%	3.2%	2.9%	1.5%
Q1 2012	59.2%	22.9%	2.0%	6.3%	9.5%

Tabela 2.1- Tabela demonstrativa de evolução do *share* do mercado. [1]

Apesar de já se falar em serviços na nuvem (*cloud*)/ computação na nuvem (*cloud computing*) desde 1994, foi em 2006 que se tornou mais popular, devido à empresa Amazon ter introduzido no mercado o seu produto chamado Elastic Compute Cloud [2]. A partir de 2007, grandes empresas, tais como a Google, Microsoft e IBM têm vindo a desenvolver várias pesquisas nesta área, com o objectivo de conseguirem melhorar os produtos, recorrendo à *cloud*. Os serviços com base numa *cloud* ganharam inúmeros adeptos, pois facilitam o acesso a informações e dados [3]. Um grande exemplo disto é a Dropbox, que oferece um armazenamento onde os seus utilizadores podem colocar o que quiserem e essa informação pode ser acedida em qualquer local desde que tenham internet. As suas bases de dados são replicadas e, assim o utilizador fica protegido para uma possível quebra de funcionamento de um dos servidores. No caso da Dropbox, é necessário instalar um programa para poder aceder às suas funcionalidades ou simplesmente ir ao site oficial.

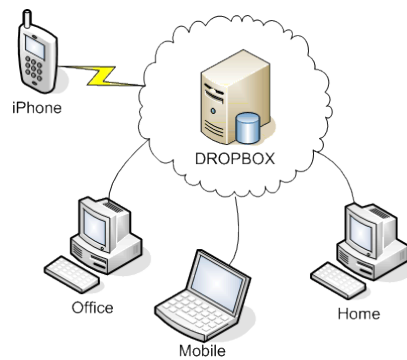


Figura 2.1 - Exemplo de uma *cloud*, em específico a Dropbox [4]

Na figura 2.1 com o exemplo da Dropbox, consegue-se verificar que todas as plataformas, independentemente do sistema operativo, conseguem ligar-se à Dropbox e ter, assim, acesso aos seus ficheiros partilhados entre todos os seus dispositivos.

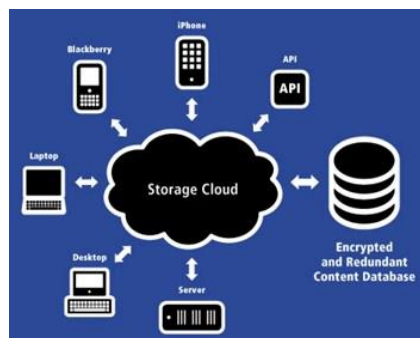


Figura 2.2 - Exemplo genérico de uma *cloud* [5]

Na figura 2.2 observa-se um possível cenário de *cloud*, que inclui: um servidor e uma base de dados encriptada e redundante. Isto significa que os dados estão encriptados e que, portanto, mesmo que tenham acesso à base de dados, não terão acesso aos dados. Existe também um mecanismo de redundância em que a informação fica armazenada em vários discos, o que significa que mesmo que um disco se danifique, essa informação estará presente noutro disco. Isto faz com que haja uma possibilidade mínima de perdas de dados. O facto de existir uma interface de programação de aplicações (API/ Application Programming Interface) faz com que seja possível aceder às várias funções a partir de diferentes sistemas operativos. Juntando todos estes aspectos, possui-se um armazenamento em *cloud*, com as características que já foram mencionadas para a Dropbox.

2.2 Sensores

Actualmente, o uso de sensores tem aumentado imenso, devido ao facto de se querer ter uma informação constante do que se está a passar, pois permite que se possa interagir com o ambiente que nos rodeia de uma forma flexível. Caso sejam operações pré-definidas, apenas são realizadas aquelas tarefas de forma repetida. Colocando sensores possibilita, por exemplo, ligar uma ventoinha se a temperatura estiver demasiado elevada, caso seja essa a intenção desejada na programação. O facto de adicionar sensores a um sistema permitirá uma maior envolvimento com o meio, levando a um aumento de capacidade em termos de resolução de problemas e de monitorização.



Figura 2.3 - NTX com quatro sensores ligados aos inputs [6]

Existem vários tipos de sensores no mercado, tais como:

- Sensores de distância que medem o espaço físico entre um ponto de referência (normalmente outro sensor) e os objectos no campo de actuação do sensor
- Sensores de proximidade, podendo verificar se está a detectar a presença de algum objecto
- Sensores indutivos capazes de detectar materiais metálicos, pois baseiam-se na variação de indutância
- Sensores capacitivos, que têm a capacidade de detectar materiais sólidos e líquidos, pois são baseados na mudança de capacitância induzida nas superfícies que se aproximam do

sensor

- Sensores de proximidade ópticos, como se pode ver na figura 2.3 e cujo modo de funcionamento é semelhante aos anteriores, na medida em que detectam a proximidade dos objectos através da propagação de uma onda desde o emissor até ao receptor
- Sensores de toque, tal como o nome indica, verificam se existe contacto e podem ser divididos em binários e analógicos. Os binários dizem se está ou não a fazer contacto e os analógicos indicam a força total aplicada [7].

Estes são alguns dos mais comuns, mas no entanto existem outros tipos de sensores, tais como temperatura, pressão, fluxo, biossensores, gás, químicos, aceleração, cmos, humidade, luminosidade, ultrasons, etc. [8].

2.3 Redes em malha

É necessário escolher a topologia que será usada, sendo portanto, necessário definir o esquema de como a comunicação vai ser realizada, incluindo os nós e ligações entre eles. Existe as seguintes topologias à disposição, tal como se pode verificar na figura 2.4: barramento, anel, estrela e malha.

A topologia barramento é constituída por uma ligação única, onde todos os dispositivos são ligados a esta. Quando um dispositivo tenta comunicar com outro, envia uma mensagem para o barramento. No entanto, se o dispositivo a quem se destina a mensagem estiver no fim do barramento e essa ligação for cortada, a informação não alcançará o mesmo.

A topologia em anel também tem alguns problemas, pois basta que uma ligação falhe e o resto do anel fica comprometido, pois a informação é transportada de forma circular.

A topologia em estrela tem como base um dispositivo central que, por norma, é um *switch/ hub*. Caso algum dispositivo falhe nada acontece ao resto da rede, no entanto caso o dispositivo central avarie, a rede não funciona.

A topologia em árvore inclui várias topologias em estrela ligadas em modo barramento [9].

A topologia em malha é um pouco mais complexa que as anteriormente descritas. Nesta topologia, cada dispositivo tem um algoritmo que reencaminha todas as mensagens para o dispositivo que estiver mais perto, com mais fiabilidade ou com o caminho mais favorável, tendo em conta o algoritmo pré-estabelecido. Foi com base nestes dados que o colega David Filipe Serra Henriques na dissertação com o tema Sensores em “movimento” escolheu a topologia em malha, pois era a que melhor satisfazia as necessidades de fluxo de mensagens.

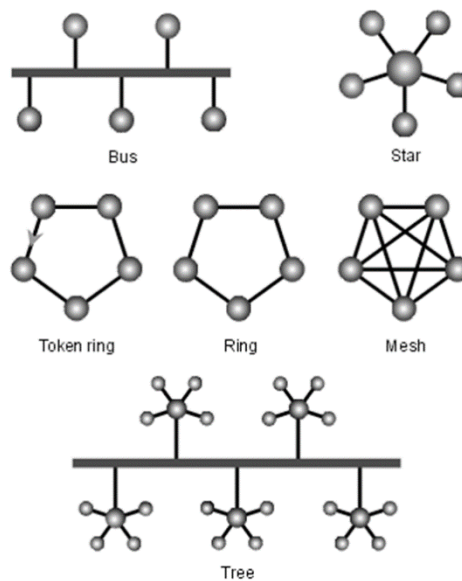


Figura 2.4 - Diferentes tipologias [10]

2.4 Segurança

Cada vez mais as pessoas recorrem à colocação dos seus dados na internet. Torna-se, assim, necessário que tal informação dos utilizadores esteja sempre a salvo, tanto em termos de privacidade, como de encriptação de dados, para não serem facilmente identificados.

Uma das formas do utilizador ficar protegido, tal como os seus dados, é usar protocolos de segurança / encriptação.

Há vários protocolos de segurança, dos quais se destaca o SSL (Security Socket Layer) e o TLS (Transport Layer Security) que basicamente é um tipo protocolo de segurança que cria uma ligação entre duas máquinas através da internet ou na rede local usando encriptação. O protocolo SSL é tipicamente usado quando um navegador necessita de se conectar de forma segura, através da internet que por si é insegura. O sistema de segurança SSL não se destina apenas a ser utilizado no navegador, pois é independente do protocolo utilizado, podendo ser utilizado em http, ftp, pop ou imap.

Tecnicamente, o SSL é um protocolo transparente, que necessita de pouca intervenção do utilizador final, desde o momento que é estabelecida uma sessão segura, usando uma porta de acesso diferente da normal (80) [11]. Por exemplo, quando se acede a um site e se verifica que no endereço está https e um cadeado ao lado significa que o website está protegido, logo os dados também.



Figura 2.5 – Descrição de um endereço web [12]

A GlobalSign foi o primeiro fornecedor de SSL a oferecer vários tipos de certificados, validação aumentada, validação de organização e validação de domínio [12], tal como se verifica na figura 2.5. A segurança trazida pelo SSL baseia-se em dois conceitos: a encriptação da informação e a autenticação. Tal como já foi referido na encriptação, os dados são tornados incompreensíveis, excepto entre o utilizador e o servidor no qual o site está alojado. A encriptação é a base da integridade e da confidencialidade dos dados. Em relação à autenticação de um certificado, é necessário que se proceda à criação de um par de chaves numéricas. Isto significa que existirá uma chave privada e uma chave pública. A chave privada é instalada no servidor, e é esta chave que cria o certificado de autenticação para o site. A chave pública é referente à outra parte do certificado SSL que, por sua vez, também é instalada no site. Ela permite aos visitantes do site em questão a encriptação da sua informação. Isto é bastante vantajoso para o utilizador do site, pois há sempre a necessidade de usar dados sensíveis (números do banco, dados pessoais, etc.). Os dados são encriptados antes de serem enviados com a chave pública. A chave privada é a única chave capaz de descriptar a informação encriptada com a chave pública.

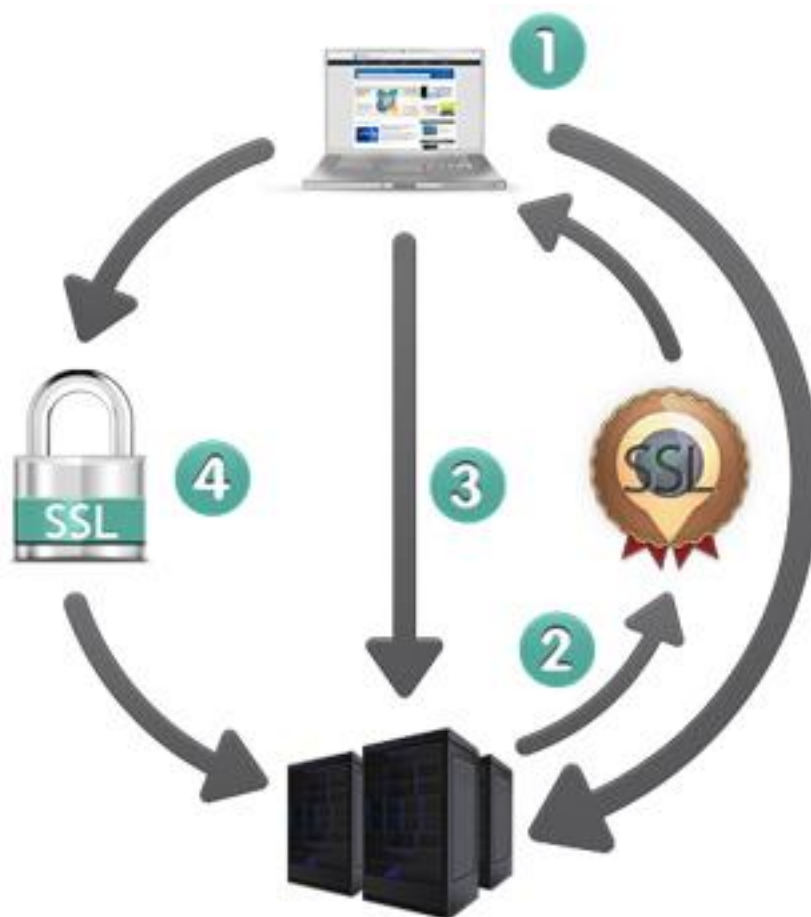


Figura 2.6 - Esquema de interacção com o SSL [13]

Na figura 2.6, o número 1 representa o pedido de inicialização de uma ligação segura via SSL ao servidor. No número 2 é apresentado um certificado ao site onde são verificadas a validade e a assinatura efectuada por um terceiro de confiança. No passo número 3 é onde é transmitida uma chave de encriptação única, cuja transmissão foi codificada com a chave pública do servidor. No passo 4 trata-se da descriptação da chave de encriptação para o servidor com a ajuda da chave privada e, assim, é estabelecida uma ligação segura [13].

Este protocolo é por exemplo usado nos websites dos bancos onde se fazem transacções monetárias, em sistemas de *login*, onde exista informação importante, em clientes de email e em muitos outros. Todas estas aplicações requerem que a informação transmitida, através da Internet, seja confidencial, ou seja, os seus números de cartão de crédito, contas de acesso a sites, palavras-passe ou informação pessoal não seja exposta na internet. A informação tem de permanecer íntegra, isto significa que, por exemplo, quando um utilizador faz uma transacção para outra pessoa, é necessário garantir que a informação não é tocada, impedindo, assim, que pessoas com más intenções consigam interceptar a informação e alterar o montante e a pessoa a quem se destina a transacção. Necessita que o certificado seja autenticado, perante uma

identidade de confiança perante os consumidores e, por fim, será necessário verificar se cumpre os regulamentos nacionais ou internacionais sobre privacidade de dados, segurança e integridade. Existe também a possibilidade de fazer a encriptação de dados e mesmo que seja interceptada por alguém com más intenções não conseguirá fazer nada, pois não consegue perceber o conteúdo da mensagem.

Apesar de já existir rádio há mais de 100 anos, a maioria da população usava telefones com cabo. Apenas nos últimos 30 anos se verifica uma tendência para o uso de telefones sem fios. Tendo em conta este facto foi necessário existir uma evolução nesta área.

Mas, no entanto, não se pode esquecer que as comunicações sem fios são mais vulneráveis do que as com fios, pois é muito mais fácil ter acesso ao meio partilhado, bastando para isso arranjar um dispositivo com capacidade sem fios para ter acesso à informação transmitida por outro dispositivo [14].

Para diminuir a possibilidade de ataques a uma rede sem fios, pode utilizar-se métodos de autenticação, de encriptação ou ferramentas de detecção de intrusos.

A autenticação é usada para impedir que um dispositivo externo à rede partilhe informações com o ponto de acesso. É usada uma chave conhecida apenas pelo cliente e pelo ponto de acesso de forma a verificar a autenticidade dos dados. A palavra-chave que se utiliza nas transmissões nunca é enviada nas comunicações para evitar que a mesma possa ser roubada.

As normas do IEEE (Institute of Electrical and Electronics Engineers) consideram dois tipos de autenticação: com chave partilhada e com chave aberta. Se for com chave partilhada, o mecanismo de autenticação baseia-se numa acção de pergunta/ resposta. Logo a seguir a ter identificado um ponto de acesso que tenha características que permitam estabelecer uma ligação, o cliente inicia a autenticação. Para isso é necessário uma trama de resposta ao ponto de acesso, fazendo com que este responda com um pacote de desafio que consiste em dados não encriptados. Após receber o pacote de desafio, o cliente encripta os dados com a sua chave e envia de volta ao ponto de acesso. Este irá verificar se os dados estão correctamente encriptados e envia novamente uma resposta de encriptação. Caso a resposta seja positiva, o cliente tem acesso à rede sem fios. Se for utilizado o método com chave aberta, tal como no método anteriormente mencionado, assim que se identifica um ponto de acesso compatível, o cliente envia um pedido de autenticação, mas o ponto de acesso não responde com um pacote de desafio, este assume que o cliente pode comunicar. Se se usar um método de encriptação, os dados enviados pelo cliente para o ponto de acesso são encriptados, onde irá ser desencriptada. Se a chave for diferente, o processo falha.

A encriptação é usada para disfarçar a mensagem. Tal como já foi mencionado, mesmo que o atacante apanhe a mensagem, este não a consegue ler sem a palavra-chave que encriptou a

mensagem. Tal como na autenticação, a palavra-chave nunca é transmitida. A encriptação de dados pertence a uma área científica de computação, designada de compressão e codificação de dados. De salientar que a codificação não se limita apenas a codificar, também tem outros objectivos, tal como, a análise e detecção de erros [15] [16].

A última forma de segurança que se mencionou foi as ferramentas de detecção de intrusos, que consiste na utilização de aplicações de detecção de intrusos, tal como o nome indica, em inglês apelidado de IDS (Intrusion Detection System).

É certo que os mecanismos de defesa foram evoluindo ao longo do tempo para dar resposta aos novos tipos de ataques que iam aparecendo. Isso traduziu-se em algumas soluções, mas, tendo em conta que foram as primeiras que apareceram, apresentavam muitas vulnerabilidades, o que faziam com que fosse impossível utilizá-las como mecanismo de segurança. A primeira solução a ser implementada foi o WEP (Wired Equivalent Privacy), a qual apresentou vários problemas, seguiu-se o EAP, que foi proposto pela Cisco, o WPA (Wi-Fi Protected Access) e o WPA2 onde ambos procuravam melhorar a solução proposta inicialmente.

O WEP foi o primeiro mecanismo de segurança a ser utilizado no protocolo 802.11. No entanto, este mecanismo de segurança foi quebrado em 2001, tornando-se assim muito inseguro.

Para tentar colmatar estas falhas de segurança, alguns fabricantes introduziram novas funcionalidades no WEP original. No entanto continuavam sem ter o resultado desejado.

Uma dessas primeiras tentativas foi usar um SSID (Service Set ID), que não é nada mais que o nome da rede sem fios, como mecanismo de segurança. Consistia em ter conhecimento prévio do SSID e o processo começaria por o ponto de acesso mandar uma trama com a informação de configuração, onde se incluía o SSID. O cliente em seguida ligava-se ao ponto de acesso tendo em conta o respectivo SSID. Após uma fase inicial em que dava a conhecer o seu SSID, este deixaria de enviar tramas com essa informação. A intenção era, caso o atacante não apanhasse a fase inicial onde era dado o SSID, este não teria acesso a ele, mas, no entanto, bastava ao atacante passar-se por um novo cliente que seria necessário informá-lo do SSID. Para isso bastava enviar uma mensagem com o SSID nulo e o ponto de acesso responderia com o seu SSID. Sem dúvida que era um sistema pouco seguro.

Havia também a possibilidade de utilizar um mecanismo de filtragem de Endereços MAC (Media Access Control). O Endereço MAC é um endereço físico que pertence à interface de comunicação, que conecta um dispositivo à rede. O processo consistia em configurar os pontos de acesso com determinados Endereços MAC, e portanto, ignorando todos os outros que não estivessem na lista de permitidos. O problema é que o atacante podia alterar o seu endereço MAC, para um valor que pertencesse à lista, ultrapassando assim o mecanismo de segurança. Para determinar um

endereço MAC válido, seria apenas necessário ler os pacotes enviados para o ponto de acesso e ler o endereço MAC.

Mas, no entanto, estas técnicas aliadas a novas tecnologias continuam a ser utilizadas com algumas modificações.

Existe também o EAP (Extensible Authentication Protocol), que é um processo de autenticação de acesso a uma rede. Os princípios do EAP assentam em flexibilidade, para assim permitir diferentes métodos de autenticação. Este foi criado na sequência dos protocolos de autenticação PAP e CHAP, que são usados no protocolo PPP, (Cap. 10 [16]), pois estes tinham problemas de segurança.

Também se tem de considerar o WPA, que surgiu devido ao IEEE demorar a produzir uma norma de segurança (802.11i) que resolvesse os problemas do WEP. De forma a tentar criar uma norma de segurança funcional, a Wi-Fi Alliance criou um subconjunto do 802.11i designado de WPA. Actualmente já existe a WPA2.

A norma 802.11i conclui o processo do desenvolvimento de uma norma de segurança baseada no WPA. Esta norma incluiu uma troca dinâmica de chaves, encriptação forte e uma autenticação de utilizador. No entanto, devido às imensas diferenças para com as anteriores, esta não é retro compatível. Para além de utilizar chaves maiores, utiliza o método de encriptação AES. [16]

A TKIP (Temporal Key Integrity Protocol) é também um método de encriptação. O TKIP disponibiliza uma chave por pacote que junta a integridade da mensagem e um mecanismo de reenvio de chave, onde a cada envio essa chave muda.

Existe a encriptação AES (Advanced Encryption Standard), também conhecida por Rijndael. Esta encriptação trabalha sobre um conjunto de bits de tamanho fixo, denominados de blocos recorrendo a uma transformação invariável que é ditada por uma chave simétrica (ou seja há uma chave pública e uma privada). Estes blocos podem ter o tamanho de 128, 168, 192, 224 ou 256 bits. O AES permite chaves de 128, 192 e 256 bits e é este tamanho que lhe dá o nome, respectivamente AES-128, AES-192 e AES-256. É considerado como sendo o sucessor do DES (Data Encryption Standard). O AES veio ocupar o lugar de TKIP [17] [18] [19].

Por norma o TKIP está relacionado com o WEP e o AES está relacionado com o WPA/WPA2, mas, no entanto, dá para escolher, tal como se verifica na figura 2.7.

WIRELESS NETWORK SETTINGS

Enable Wireless : ☒

Wireless Network Name : Lima2 (Also called the SSID)

Enable Auto Channel Selection : ☐

Wireless Channel : 6

Transmission Rate : Best (automatic) (Mbit/s)

WMM Enable : ☒ (Wireless QoS)

Enable Hidden Wireless : ☐ (Also called the SSID Broadcast)

WIRELESS SECURITY MODE

Security Mode : Enable WPA/WPA2 Wireless Security (enhanced)

WPA/WPA2

WPA/WPA2 requires stations to use high grade encryption and authentication.

Cipher Type : AUTO(TKIP/AES)

PSK / EAP : PSK

Network Key :

(8~63 ASCII or 64 HEX)

Figura 2.7 - Configuração de um encaminhador

Pode-se verificar vários aspectos nesta figura 2.7, que demonstra a configuração possível num encaminhador. Por exemplo, onde diz *Enable Hidden Wireless* significa que, activando esta opção, o sinal wireless, apesar de estar a ser difundido, não é visível a quem procura as redes, fazendo com que apenas pessoas com o nome certo da rede possam aceder. No campo *Wireless Security Mode* é onde se escolher o tipo de protecção que se quer, podendo-se escolher entre não ter segurança, o que é desaconselhável por razões óbvias e entre ter a protecção WEP, o que também já não é aconselhável devido a já ter sido quebrada a segurança. Resta apenas o modo de segurança WPA/WPA2 que é o mais usual e aconselhado.

No campo *Cipher Type* (tipo de cifra) pode-se escolher qual a encriptação desejada, podendo escolher entre AES e TKIP.

O tipo de autenticação pode alternar entre PSK (que significa pre-shared key, e como o nome indica, foi uma chave previamente partilhada entre os dois dispositivos usando uma forma segura de comunicação) ou EAP.

É necessário mencionar a RSA, pois foi uma das que melhor foi implementada até hoje, em termos de algoritmo de criptografia e assinatura digital, tornando-se assim essencial falar nela. RSA deve-se ao nome dos seus criadores, que eram professores do Instituto de Tecnologia de Massachusetts (MIT), Ronald Rivest, Adi Shamir e Leonard Adleman, a primeira letra do apelido de cada Professor deu o nome à encriptação. São também fundadores da actual empresa RSA Data Security Inc., sendo a mais bem sucedida implementação de sistemas de chaves assimétricas. É certo que por vezes a segurança foi quebrada, mas rapidamente fizeram um novo algoritmo com mais bits de encriptação. O RSA é basicamente o resultado de dois cálculos

matemáticos, um para encriptar e outro para desencriptar, sendo que usa duas chaves criptográficas, uma pública e uma privada. Tal como já foi mencionado, a chave pública é usada para encriptar a mensagem e a chave privada para desencriptar a mensagem [17].

A Secure Hash Algorithm, mais geralmente apelidada por SHA, tem várias vertentes, entre as quais o SHA-0, SHA-1, SHA-2, SHA-3. O SHA-0 já se encontra bastante desactualizado e praticamente já caiu em desuso, o SHA-1 é uma função de *hash* (tem como função criar uma mensagem de tamanho fixo, através de uma cadeia de caracteres de qualquer tamanho) que foi pensada pela Agência de Segurança Nacional dos Estados Unidos, que produz um valor de dispersão que é normalmente tratado como um número hexadecimal de 40 dígitos, onde portanto o tamanho de saída é 160 bits. No SHA-2 existem várias versões: a SHA-224, SHA-256, SHA-384, SHA-512, que produzem uma saída respectivamente de 224, 256, 384, 384 e 512 bits. Para escolher o SHA-3 foi necessário fazer uma competição pública para projectar um novo algoritmo de hash que iria substituir o SHA-1 e SHA-2 e o algoritmo escolhido foi o de Keccak, que tem como vantagens o uso de um algoritmo mais complexo, não tem limite de tamanho de mensagem a verificar, entre outras. O SHA-3 está disponível na vertente de SHA3-224, SHA3-256, SHA3-384, SHA3-512 e SHAKE128 e SHAKE256, tendo estas últimas a vantagem da mensagem de saída ser potencialmente infinita, o que aumenta bastante a segurança e o grau de complexidade em conseguir quebrar esta segurança. Para saber mais visitar a página do criador [20].

Existe ainda o MD5 (Message Digest 5), cujos antecessores são o MD4 e MD2. É um algoritmo de hash, tal como o SHA, de 128bits, unidireccional, desenvolvido pela RSA Data Security, Inc, cujo processo é muito utilizado por programas com protocolos ponto a ponto (p2p) para verificar a integridade do arquivos/dados, pois assim consegue-se verificar se os dados originais foram alterados [17] [18].

Há muitos outros tipos de seguranças, autenticações e formas de protecção que não foram aqui mencionadas. Apenas se referiu aquelas que foram tidas em consideração para o projecto, e que, de alguma forma, já estejam implementadas no que foi utilizado e que se julga ser importantes apresentar.

2.5 Projectos semelhantes

Importa agora referir alguns projectos semelhantes ao que se está a desenvolver, pois actualmente todas as pessoas pretendem ter informação sobre o meio que as rodeiam, sejam essas informações provenientes de sensores ou outros dispositivos.

A Link tem um protótipo de redes de sensores, representado na figura 2.8, denominado HealthCare e tem como objectivo o acompanhamento remoto do paciente, maximização de recursos nos hospitais, otimizar cuidados na área da saúde, monitorizar eventos de alarme (por exemplo, fogo). Se um paciente utilizar vários tipos de sensores para monitorização de sinais vitais/ índice de actividade física, a informação recolhida seria analisada num servidor central e o sistema forneceria uma sensação de proximidade ao utilizador e monitorização constante do seu médico/familiar.

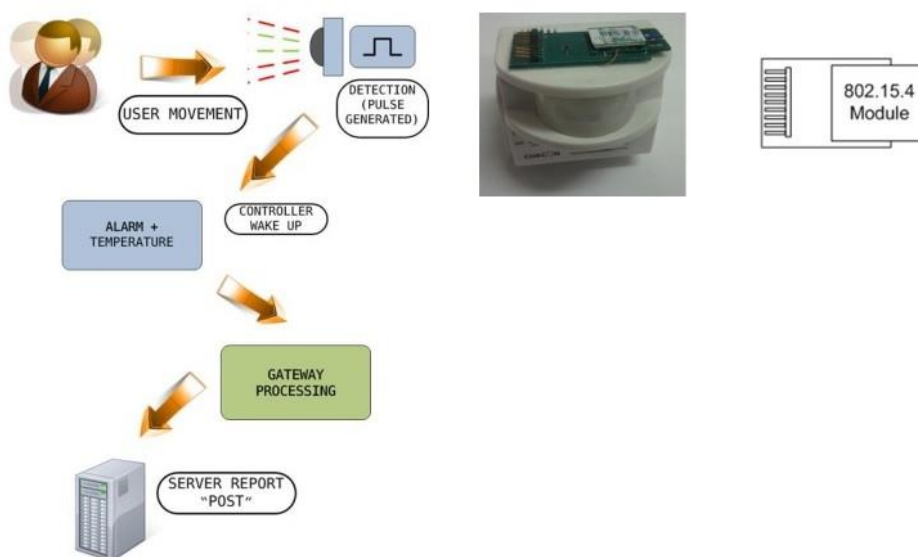


Figura 2.8 - Representação do esquema protótipo da Link [21]

Todos os sensores usam baterias e com o objectivo de maximizarem a autonomia das baterias, recorreram à norma IEEE 802.15.4, devido às suas baixas necessidades energéticas. Imaginando que a pessoa tem os sensores no seu corpo e vai para o exterior, deixa de poder comunicar com o *gateway*. Neste caso, o sensor ao qual foi adicionado capacidade de comunicação, armazena os dados e assim que voltar a encontrar o *gateway*, envia os dados. No caso deste projecto, como o *gateway* é um *smartphone*, este problema não existiria [21].

Existe também a Valarm, que através de uma ligação USB OTG recolhe os dados dos sensores que estão ligados por cabo, tal como se pode verificar na figura 2.9. Logo aqui, constata-se uma desvantagem, pois é necessário ter uma ligação por fios. Se existirem sensores em diferentes pontos, é necessário um telemóvel em cada ponto, o que é bastante desvantajoso. Pode-se enviar os dados, através de Wi-Fi, caso se tenha um ponto de ligação por perto, por Bluetooth, sendo para isso necessário ligar-se a outro telemóvel ou enviar através de redes móveis [22].



Figura 2.9 - Exemplo de configuração do Valarm [23]

A empresa Libelium tem um produto designado de waspmote. Este tem a capacidade de comunicar através de IPV6, ou melhor, 6LoWPAN (IPv6 over Low power Wireless Personal Area Networks) [24]. Para efectuar a recolha de informação é necessário adquirir o Waspote Plug & Sense e posteriormente o envio para a *cloud* recorre ao Meshlium. O Meshlium poderá enviar a informação para *clouds* previamente definidas através de cabo de ethernet, Wi-Fi ou 3G/ GPRS. Em caso de falha de envio a informação será guardada numa base de dados interna tal como se verifica na figura 2.10 [25]. Este produto implica que sejam comprados vários equipamentos proprietários à parte, aumentando o valor do produto final.

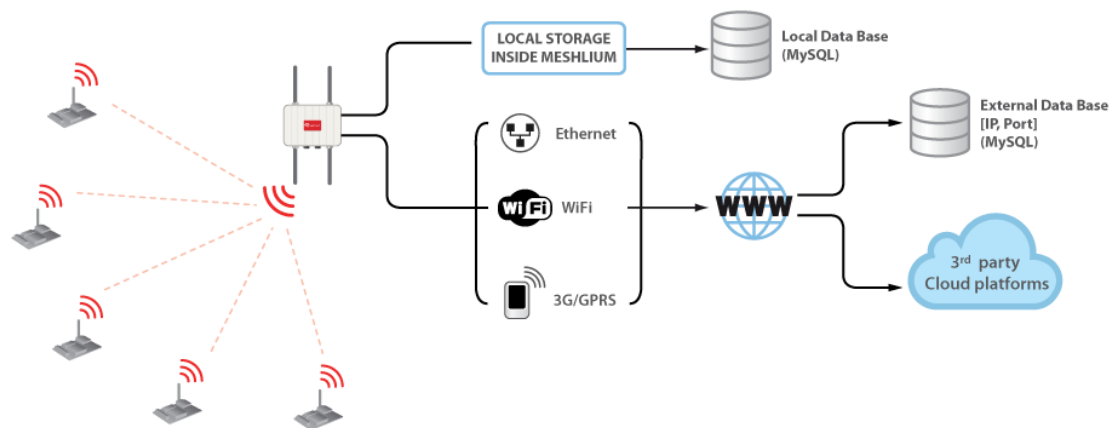


Figura 2.10 - Arquitectura do produto da empresa Libelium [25]

O SensorCloud, feito por LORD MicroStraing, tem um conceito semelhante ao proposto nesta dissertação, tal como se pode verificar na figura 2.11, dado que utiliza um gateway que é um retransmissor que possibilitará a transmissão até 2 km. Mas, no entanto, implica também um ponto de internet e necessita que seja feito o pedido, através de uma página à qual só se consegue aceder após se estar conectado ao retransmissor, sendo necessário de carregar no botão de enviar os ficheiros, que são recebidos em formato excel [26].



Figura 2.11 - SensorCloud [26]

Existem muitos outros que não foram mencionados. Os supramencionados apenas serviram para fazer a comparação entre este projecto e o disponível no mercado.

2.6 Tecnologias Sem fios *versus* com fios

Os tipos mais comuns de tecnologias com fios vocacionadas para redes são LANs (Local Area Networks), MANs (Metropolitan Area Networks) e WANs (Wide Area Network).

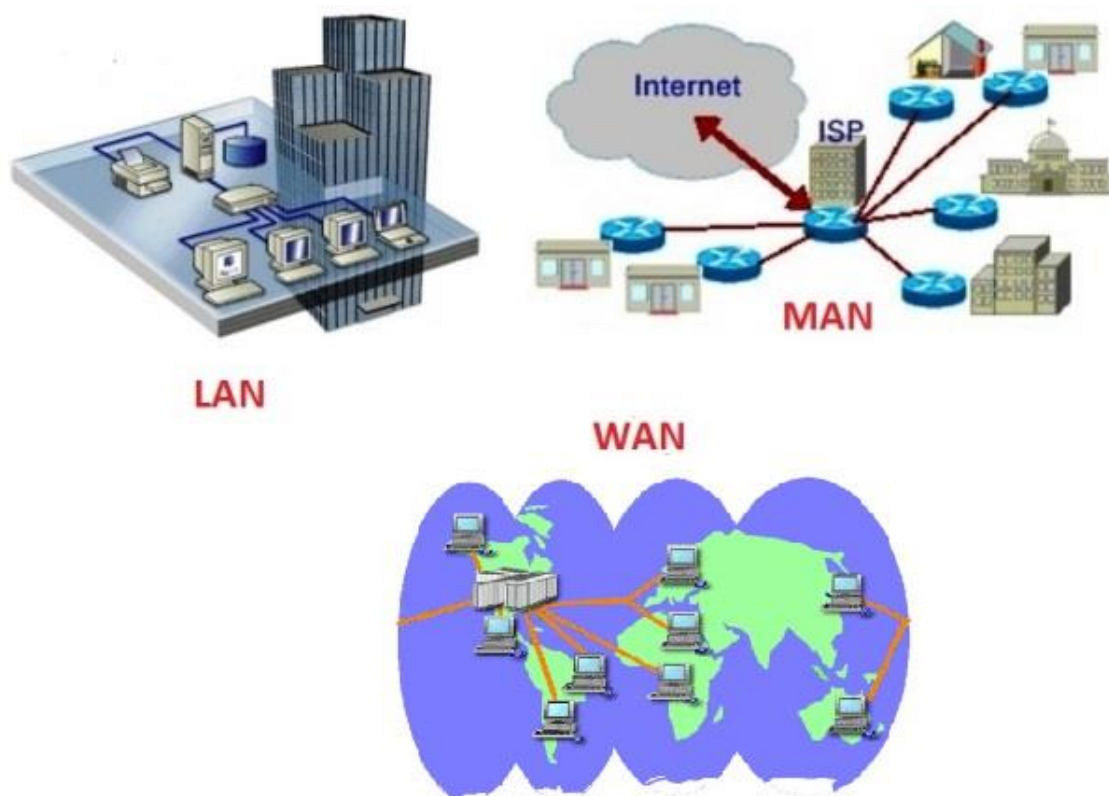


Figura 2.12 - Imagem demonstrativa dos três tipos de rede [27] [28]

Como se pode verificar na figura 2.12, uma rede LAN consiste na interligação de um edifício (portanto, uma pequena área), tem tipicamente taxas de transmissão bastante superiores às outras e é usado nos casos em que é vantajoso partilhar a informação com outras pessoas de forma rápida, como é o caso de empresas.

Uma rede MAN interliga várias LAN, que acontece numa cidade, por exemplo. A ligação, sendo de fibra óptica, fornece uma boa velocidade de comunicação entre todas as redes. O conceito de MAN é muito similar ao de um ISP (Internet Service Provider), onde todos os clientes se ligam ao ISP e podem partilhar informação entre si, caso desejem. A grande velocidade que a LAN pode alcançar proporciona um grande número de vantagens, que motivam a implementação à escala metropolitana. O principal mercado da MAN é o cliente que necessita de grandes capacidades numa área metropolitana. A MAN pretende dar um serviço de grande qualidade, a

baixo preço e ter uma melhor eficiência em relação à obtida pelos serviços através da linha telefónica.

Tal como já referido, a rede WAN consiste tipicamente em interligar redes LAN, metropolitanas e equipamentos de rede, numa grande área geográfica, por exemplo, um país ou um continente. Se for considerado que se interligam todos os países do mundo, então o exemplo de uma WAN é a internet [29].

É de acrescentar que se aplicam as mesmas topologias utilizadas no capítulo redes em malha, onde muda apenas o equipamento usado.

O transporte dos dados, neste tipo de redes, pode ser feito maioritariamente por três materiais: cabo coaxial, par trançado, vulgarmente chamado cabo *ethernet*, e fibra óptica. “Coaxial - Constituído por um fio de cobre condutor revestido por um material isolante e com blindagem. É bastante eficaz para transmissões com frequências elevadas, mas não é tão eficaz nas velocidades de transferência de dados. Em termos de rede de dados, permite velocidades de 10 Mbps e distâncias até 500 metros;

Par Trançado - Formado por 4 pares de fios (total de 8 fios) trançados par a par. Podem ser de 2 tipos: UTP que não é blindado e o STP que é blindado. Existem 4 categorias principais deste cabo: Cat-3 - frequência de 16 MHz e velocidade até 10 Mbps; Cat-5 - frequência de 100 MHz e velocidade até 100 Mbps; Cat-5e - frequência de 125 MHz e velocidade até 1 Gbps (Giga bit por segundo); Cat-6 - frequência de 250 MHz e velocidade até 1 Gbps; Cat-6a - frequência de 500 MHz e velocidade até 10 Gbps. A distância é de 100 metros, exceto na categoria 6a, em que para 10 Gbps a distância é de 55 metros.

Fibra Ótica - Caracteriza-se por ter um núcleo de fibra de vidro ou acrílico e transmitir pulsos de luz, ao invés de sinais elétricos dos cabos anteriores. A fibra ótica só transmite numa direção, pelo que os cabos são aos pares (um para enviar e outro para receber). Existem dois tipos de fibra ótica:

Multimodo - núcleo mais grosso, logo com menor desempenho, pois a luz reflete nas paredes do núcleo. Distância até 2 Km;

Monomodo - núcleo mais fino, logo com maior desempenho. Distância até 3 Km;

De acordo com testes laboratoriais, a velocidade pode chegar até 200 Gbps.” [28].

Entretanto, os valores por cabo coaxial sofreram alterações, devido a uma nova norma que permitiu maiores velocidades. A fibra óptica já envia e recebe em simultâneo, recorrendo para isso a uma modulação em frequência.

2.6.1 Tipos de comunicação sem fios

Wimax

Wimax é a sigla para Worldwide Interoperability for Microwave Access (Interoperabilidade mundial para acesso por micro-ondas). Wimax é a próxima geração, em termos de comunicações de redes sem fios. Conta com a semelhança em relação ao Wi-Fi, mas com a excepção de oferecer acesso de alta velocidade, numa área muito mais ampla, aliado ao facto de ter menos interferências. Se os portáteis e dispositivos com acesso à internet estiverem habilitados ao serviço Wimax, poderão aceder à internet, onde não haja pontos de acesso Wi-Fi, desde que estejam dentro da área de cobertura. Estas estações podem cobrir até 50 km, enquanto as estações de redes móveis apenas cobrem entre 5 a 15 km. Esta tecnologia baseia-se na norma IEEE 802.16 para WMANs [30].

Wi-Fi

A primeira norma do grupo 802.11 a ser aceite pela indústria foi a 802.11b, alcançando uma velocidade máxima de 11 Mbps em 400 metros em ambiente aberto ou 50 metros em lugares fechados. Em 1991 foi lançada a norma 802.11a e 802.11b, no entanto, tal como já foi referido, a 802.11b foi a que conquistou a indústria, devido a usar a frequência 2.4Ghz, ao invés de 5Ghz, como a 802.11a (apesar de sofrer menos interferências, em alguns países esta frequência não se pode usar). Apesar dos produtos 802.11b serem baseados na mesma norma, há sempre uma preocupação para que os produtos de diferentes fabricantes possam comunicar entre si. Para ajudar nesta questão, foi criado um consórcio em 1999, chamado Wireless Ethernet Compatibility Alliance (WECA). Esta organização foi posteriormente renomeada para Wi-Fi (Wireless Fidelity) Alliance e tratava de certificar que os produtos 802.11b eram compatíveis entre si [29].

Esta tecnologia foi evoluindo passando pela norma 802.11g, sendo disponibilizada em 2003 e considerada a sucessora da norma 802.11b. Era compatível com a versão anterior e portanto poderiam comunicar entre si, estando apenas limitados à velocidade da norma mais lenta. A 802.11g tinha taxas de transmissão máximas de até 54 Mbps, tendo praticamente a mesma cobertura da norma anterior.

A 802.11g evoluiu de forma oficial para a 802.11n em 2009 e com ela trouxe alguns melhoramentos. Tem uma característica chamada Multiple-Input Multiple-Output (MIMO), que possibilita o aumento considerável das taxas de transferências de dados, recorrendo à emissão através de várias antenas. Usando o modo MIMO consegue chegar teoricamente até 600 Mbps e, usado de forma normal, tem velocidade máxima de 150 Mbps.



Figura 2.13 - Exemplo de indicação de um encaminhador

Tal como se verifica na figura 2.13, o encaminhador que usufruiu da norma 802.11n pode trabalhar com duas frequências, a 2,4 e 5 Ghz, podendo, assim, tirar partido do facto da banda de 5 Ghz não estar saturada [31].

A 802.11ac, apesar de já ter sido introduzida em 2012, ainda não está implementada na maioria dos encaminhadores, devido a tornar o produto mais caro, sendo que a maior parte deles continua com a norma n. Mas a norma ac oferece até 1733 Mbps na frequência dos 5 Ghz, o que é uma grande evolução. De referir que esta norma também é compatível com a sua antecessora norma n [32].

Todas estas normas têm segurança, cujo assunto foi abordado no capítulo de segurança.

De alertar que existem outras normas, mas que não se irá falar nelas, devido a não serem muito usadas no país.

Recorrendo à tecnologia Wi-Fi desenvolveu-se uma nova forma de comunicação entre dispositivos, apelidada de Wi-Fi Direct. Desde que os dois dispositivos intervenientes tenham uma placa de *wireless* compatível com Wi-Fi Direct, poderão trocar dados facilmente.

NFC

NFC ou Near Field Communication permite interações bidireccionais de forma simples e segura entre dispositivos que tenham essa tecnologia, permitindo assim efectuar transacções sem qualquer tipo de contacto, aceder a conteúdo digital e ligar-se a dispositivos com um único toque. O NFC possibilita a troca de informação a uma distância inferior a 4 centímetros, com uma velocidade máxima de 424kbps [33].

À semelhança das redes com fios já mencionadas, existem essas mesmas redes, mas sem fios. WLAN, WMAN e WWAN, o conceito mantém-se, apenas muda o facto de ser sem fios.

Bluetooth

O Bluetooth insere-se na categoria de WPAN (Wireless Personal Area Network), usando a norma 802.15, iniciando a sua história em meados de 1994. O conceito que está subjacente ao Bluetooth é disponibilizar uma rede de pouco alcance, usando a frequência de 2.4 Ghz, que está disponível globalmente, para uso de pequenas potências, sem ser necessário licenciar. A primeira versão (1.0) teve a capacidade de trocar informação entre dois dispositivos equipados com tecnologia Bluetooth até 720kbps, numa distância de 10 metros. O Bluetooth foi feito com a intenção de suportar uma grande lista de aplicações, entre elas dados, áudio, imagens ou mesmo vídeo. Por exemplo, é possível fazer chamadas, através de auriculares sem fios ligados por Bluetooth a um telemóvel, eliminar os cabos usados para ligar computadores a impressoras, teclados e ratos, entre muitos outros exemplos.

O Bluetooth foi concebido para trabalhar em ambientes de muitos utilizadores. Até oito utilizadores (um *master* e sete *slaves*) podem comunicar numa pequena rede chamada piconet. Podem existir até dez redes piconet na mesma zona. Para garantir que a segurança é assegurada, cada ligação é codificada e protegida contra espionagem e interferências [29].

O Bluetooth foi evoluindo até a versão 1.2 sem mudanças nas velocidades, apenas corrigindo erros e melhorando a sua estabilidade. Na versão 2.0 normal conta com a mesma velocidade das anteriores, mas se tiver o padrão EDR (Enhanced Data Rate) a velocidade sobe para 2.1 Mbps efectivos, diminuição do consumo de energia, correcção de erros e melhor comunicação entre dispositivos. Na versão 2.1 EDR houve melhorias de consumo, melhor segurança e mantendo a velocidade da versão anterior. Em 2009 saiu uma nova versão, a 3.0 HS, que teve como principal novidade as grandes velocidades de transmissão. Os dispositivos com esta versão poderão atingir os 24 Mbps, aliado a um controlo inteligente do gasto de energia. Mesmo tendo uma grande evolução o Bluetooth 3.0 continua a ser retro compatível. No entanto, se estiver ligado a uma versão mais antiga, trabalha à velocidade do dispositivo mais lento. De acordo com diferentes fontes de informação, no final de 2009, início de 2010, foi lançada a versão 4.0, também chamada de Bluetooth low energy. Até à data já se tem a versão 4.2, onde há melhoramentos na área de ligação IP, privacidade e velocidade [34], [35].

		ZigBee	Bluetooth technology	802.11b	802.11g	802.11a	802.11n	UWB
Throughput	Mbps	0.03	1-3	11	54	54	200	200
Max range	ft	75	30	200	200	150	150	30
Sweet spot	Mbps-ft	.03@75	1-3@10	2@200	2@200	36@100	100@100	200@10
Service	bps-ft ²	530	314M	251G	251G	1.13T	3.14T	62G
Power	mW	30	100	750	1000	1500	2000	400
BW	MHz	0.6	1	22	20	20	40	500
Spectral efficiency	b/Hz	0.05	1	0.5	2.7	2.7	5	0.4
Power efficiency ¹	mW/Mbps	1000	100	68	19	27	10	2
Power efficiency ²	mAh/GB	2211	67	46	12	18	7	1.3
TTGB	Time	3.1 day	2.2 hr	12 min	2.5 min	2.5 min	40 sec	40 sec
Price	US\$	\$2	\$3	\$5	\$9	\$12	\$20	\$7

Tabela 2.2 – Comparação entre várias tecnologias [36]

Tal como se pode verificar na tabela 2.2, o Bluetooth tem um preço inferior. Em contrapartida, tem um alcance reduzido quando comparado com o Wi-Fi.

2.6.2 Vantagens/desvantagens

Cada tecnologia tem as suas vantagens/ desvantagens inerentes, a questão aqui prende-se com a escolha pessoal, questão económica e estética e fiabilidade.

A pessoa pode simplesmente gostar do conforto de poder estar ligada à internet e não ter fios ligados a si, e, neste caso, é uma vantagem o facto de não ter fios a incomodar. A questão económica é de realçar, pois certamente o cabo é mais barato que um encaminhador sem fios, mas em seguida debatemo-nos com a questão estética, pois fica-se com um fio a passar pela casa. No entanto no que diz respeito à fiabilidade, pode-se confiar no cabo de cobre, pois tem menos possibilidade de interferências, apesar de não ser completamente imune.

2.7 Cloud computing

2.7.1 O que é?

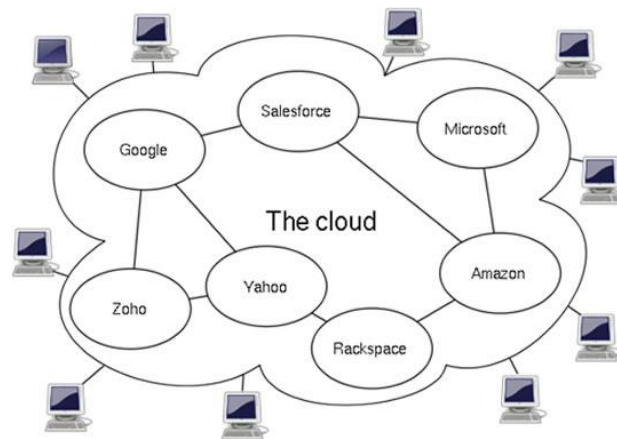


Figura 2.14 - O mundo da *cloud* [37]

O conceito da *cloud* consiste na gestão e oferta de aplicações, dados e informação como um serviço, que é fornecido através da internet. Este serviço tem à disposição memória, armazenamento e processamento, através de computadores e servidores partilhados estando interligados pela rede, tal como se verifica na figura 2.14.

A *cloud* pode escalar de várias formas, verticalmente e horizontalmente. Se escalar na vertical, pode-se ir aumentando o espaço de disco, a memória disponível, os processadores, mas, no entanto chegará a uma altura em que o limite físico do servidor é alcançado e, aí, é necessário colocar outro servidor, ao qual é chamado de escalamento horizontal.

2.7.2 Vantagens/desvantagens

A *cloud* tem a vantagem de se poder aceder a aplicações a partir da internet, sem que estas estejam instaladas, em computadores ou dispositivos físicos. Mas no entanto há outros benefícios:

- Na maioria dos casos, o utilizador pode aceder às aplicações, independentemente do seu sistema operativo ou do equipamento usado.
- O utilizador não precisa de se preocupar com o que é necessário para correr a aplicação, tais como, *hardware*, fazer cópias de segurança, controle de segurança, manutenção, etc.

- Partilhar informações e efectuar trabalho colaborativo torna-se mais fácil, pois, todos os utilizadores acedem a aplicações e aos dados no mesmo local, a *cloud*.
- Varia de fornecedor para fornecedor, mas imagine-se, que o servidor, onde a aplicação/informação se encontra alojada para de funcionar, outros servidores, que contêm a informação replicada entram em funcionamento e, assim, continuam a oferecer o serviço, contando assim com alta disponibilidade.
- O utilizador tem controlo sobre os custos, pois algumas *cloud* têm um nível grátis e apenas é necessário pagar, quando ultrapassa os recursos ou o tempo de utilização. Não é obrigatório pagar por uma licença integral de uso.
- Pode ser necessário instalar um programa no cliente, no computador ou noutro dispositivo, mas, no entanto, todo ou a maior parte do processamento, e até mesmo o armazenamento de dados é feito pela *cloud*.

Deve-se ter em atenção, que independentemente da aplicação com a *cloud*, o utilizador não necessita de conhecer toda a estrutura que existe por trás. Portanto o utilizador não precisa de saber quantos servidores executam determinada tarefa, quais as configurações de *hardware* usadas, bem como a forma como o escalamento é feito, qual a localização física de onde se encontram os servidores. O que importa é saber que a aplicação está disponível [38] [3].

Infelizmente, todas as tecnologias têm as suas desvantagens, e a *cloud* não transcende a essa questão. A *cloud*, devido à capacidade enorme que tem de processamento, pode ser usada como meio para prejudicar outras pessoas, nomeadamente, fazer sucessivos acessos a um site, de forma a deixar de funcionar. Devido também à grande capacidade de armazenamento, por vezes, é usado para difundir conteúdo, sem o consentimento dos seus autores, mas é o preço a pagar pela evolução.

2.7.3 Modelo computacional *Cloud*

Existem vários tipos de uso que se pode dar à *cloud*:

- *Software as a service* (SaaS) ou, em português, software como serviço. Trata-se de uma forma de trabalho em que o *software* é disponibilizado como serviço, assim, o utilizador não precisa de comprar licenças. Por norma, a entidade em questão paga um valor periódico, tendo em conta os recursos utilizados e/ou tempo de uso.

Para explicar melhor os benefícios do uso do SaaS, suponha-se que existe uma empresa que tem vinte funcionários e necessita de um programa para gerar facturas. Apesar de já existirem várias

soluções no mercado, poderia ser necessário a empresa comprar uma licença para poder usar o programa escolhido e em alguns casos até mesmo *hardware* para o poder correr. Isto implica altos investimentos para a empresa. Por outro lado, se se optar por utilizar um fornecedor de *software* de facturação que trabalha com base no SaaS, a sua situação poderá ser bastante facilitada. Se esse fornecedor oferecer esse serviço através de *cloud computing* e cobrar, apenas, pelo número de funcionários e/ou pelo tempo de uso, isso levará a que a empresa que contratar este serviço, tenha um menor valor a pagar pelo uso do *software*. Não é necessário estar preocupado com *hardware*, instalação, actualizações, manutenções, entre outros, pois estas tarefas ficam por conta do fornecedor. Além de que o tempo de espera, entre contratar o serviço e começar a usá-lo é extremamente baixo, devido a não ser preciso instalar algo no computador, pois basta ter acesso à internet e, no máximo, permitir que o plugin corra no navegador de internet das suas máquinas.

Para garantir de facto que uma solução seja vendida como SaaS, deve cumprir os seguintes requisitos:

- Acesso à aplicação via web;
- A gestão da aplicação é realizada de forma centralizada
- O usuário não é responsável por lidar com actualizações ou aplicações de correcções na aplicação
- A aplicação é entregue no modelo de 1 para n
- Existência de APIs para permitir integrações externas

- Platform as a Service (Paas), ou plataforma como serviço, refere-se a um tipo de solução mais ampla para aplicar em determinadas situações. Possui todos, ou quase todos, os recursos necessários ao seu funcionamento, tais como, armazenamento, base de dados, escalabilidade (aumento automático da capacidade de processamento ou armazenamento), suporte a linguagens de programação, etc.

Há uma série de maneiras diferentes para caracterizar PaaS, mas, algumas das características mais comuns encontradas em diversos fornecedores são:

- Ambiente para desenvolver, testar, implantar e manter aplicações de forma integrada e escalável, para cumprir todo o processo de desenvolvimento
- A arquitectura multi-tenant, onde vários usuários podem utilizar em simultâneo a mesma aplicação

- Escalabilidade, incluindo balanceamento de carga e faillover (processo no qual uma máquina assume os serviços de outra, quando esta apresenta falha)
- Integração com serviços web e bases de dados através de padrões comuns
- Ferramentas para lidar com facturação e gestão de assinaturas
- Segurança integrada
- Ambiente dimensionado e pronto para utilização de aplicações complexas

- Database as a Service (DaaS) , ou base de dados como serviço, tal como o nome sugere, é vocacionada para o fornecimento de serviços para armazenamento e acesso a grandes volumes de dados. A vantagem do utilizador que usufruir destes serviços é que conta com maior flexibilidade para expandir a base de dados, partilhar as informações com outros sistemas, permitir o acesso remoto a usuários autorizados, etc.

- Infrastructure as a Service (IaaS), ou infraestrutura como serviço. Idêntico ao conceito de PaaS, mas, aqui o foco é a estrutura de *hardware* ou de máquinas virtuais, tendo o utilizador acesso a recursos do sistema operativo.

As ofertas de soluções, os custos, a alta disponibilidade e o nível de profissionalização dos fornecedores são os principais factores para optar por IaaS. Deve-se contar com as seguintes características:

- Recursos são contratados como um serviço;
- Pode ter um custo variável pelo uso ou definido de forma prévia;
- Alta escalabilidade com rapidez e eficiência
- Monitoramento e gerenciamentos avançados

- Testing as a Service (TaaS), ou ensaio como serviço. Proporciona um ambiente que permite ao utilizador testar aplicações e sistemas de forma remota, simulando o comportamento destes a nível de execução [38] [39].

O PaaS, DaaS, IaaS e TaaS são derivados do SaaS, mas com outros nomes para diferenciar os seus serviços.

2.7.4 Exemplos

Já se utiliza o conceito de SaaS na globalidade de serviços de internet que se consome diariamente, por exemplo, um motor de busca (Google, Bing, Yahoo) ou o correio electrónico. Neste serviço o utilizador final não tem de se preocupar com instalação de *software*, configuração de rede, alocação do servidor, licenças para programas, etc. O fornecedor de serviço cobra uma taxa por disponibilizar o serviço e dar suporte e o software é utilizado exclusivamente através da internet. O serviço de tipo SaaS mais comum do mercado é o Google docs, gmail e sales force.

O PaaS é análogo ao SaaS, excepto que o *software* não é apresentado na internet, é uma plataforma (um ambiente) para a criação, hospedagem e controle de software. Usar o PaaS vai-se tornar uma abordagem com bastante incidência no desenvolvimento de *software*, pois tem a capacidade de automatizar processos, utilizando componentes pré-definidos, blocos pré-construídos, fazendo com que desenvolver uma aplicação seja muito mais fácil. O tipo de serviço PaaS mais comum é Google AppEngine, force.com e heroku.com da Salesforce.

2.8 Evolução dos telefones móveis/das redes móveis

Sempre se comunicou, mas não da forma como se comunica actualmente. Os nossos antepassados comunicavam por fumo, comunicação por código morse, cartas que, apesar de serem usadas há imenso tempo, ainda hoje são usadas. As enormes centrais telefónicas, onde existia uma pessoa que tinha de ligar os cabos de forma manual quando se fazia uma chamada para um telefone fixo, também já não existem. Mas tal como o nome indica, o telefone fixo tinha de estar sempre no mesmo lugar o que significaria que caso a pessoa estivesse fora de casa, estaria incontactável.

Apesar da primeira tecnologia, que permite fazer chamadas e receber fora de casa ter aparecido em 1983, e ser apelidada de AMPS o que significa Advanced Mobile Phone System, sendo a primeira geração de tecnologia para redes móveis, não teve muito impacto no mundo, pois foi mais usada nos estados unidos e cobria pouca área. Esta tecnologia utilizava FDMA (frequency division multiple access) o que significa que cada banda de frequência alocada para as comunicações de telefones móveis era dividida em sub intervalos de 30 kHz, chamados canais, onde cada canal podia suportar uma conversação. Com a FDMA, cada canal apenas pode ser associado a um utilizador de cada vez [40] [41].

O surgimento de uma tecnologia de segunda geração de telefones móveis (2G), em 1991 com o nome de GSM que significa Global Systems for Mobile Communications, foi uma grande

revolução, pois foi praticamente implementado em todo o mundo, devido às suas características, além de se ter tornado uma comunicação digital, ao contrário da geração anterior que era analógica. Tinha capacidade para suportar um maior número de utilizadores ao mesmo tempo e, por outro lado, também cobria áreas maiores. Isto levaria a que muitos países aderissem a essa tecnologia. De salientar que houve uma geração digital, referente à primeira geração do AMPS chamada Digital-AMPS. O D-AMPS foi cuidadosamente desenvolvido para coexistir com o AMPS, permitindo que os telemóveis de primeira e segunda geração pudessem operar em simultâneo. Apesar do D-AMPS e GSM serem semelhantes, pois ambos são rede de telecomunicações móveis, a multiplexagem é feita por divisão de frequência, mas no entanto, os canais do GSM são de 200kHz e os de D-AMPS são de 30kHz, o que dá uma clara vantagem ao GSM, em termos de taxa de dados, pois é superior

Tome-se como exemplo o telemóvel Nokia 3310 e o Nokia 3330, aparentemente eram iguais exteriormente, mas tinha uma grande diferença no seu interior. O 3310 possuía apenas a tecnologia GSM, em contrapartida o 3330 já tinha GPRS (General Packet Radio Service). O GPRS é uma evolução da rede GSM, e, daí, se poder considerar que é da segunda geração e meia (2.5G). Este permitiu às operadoras GSM fornecerem o acesso a redes de comutação de pacotes, como é o caso da internet.

Ainda que esta troca de pacote de dados pudesse ser paga à operadora, dava uma grande vantagem ao 3330, sobre o 3310, pois permitia-lhe visualizar imagens, ainda que cromaticamente, mas isso era uma característica intrínseca do aparelho.

A diferença de funcionamento entre GPRS e GSM está intimamente relacionada com a velocidade a que conseguem trabalhar, pois o GSM apenas trabalha a 9kbps, enquanto que o GPRS vai até aos 170kbps. Sabendo que no GSM faz sentido falar da duração de uma chamada, já no GPRS faz sentido falar na quantidade de dados que foram consumidos.

A tecnologia GSM inseriu o uso de um cartão chamado SIM (Subscriber Identity Module). Este pequeno cartão contém informação do seu número, da operadora, lista de contactos, entre outros. O facto de o cartão possuir a informação da pessoa e do serviço que contractou, fez com que se tornasse muito popular, fazendo com que, mesmo que a pessoa mudasse de telemóvel, continuava com o mesmo número e com a mesma lista de contactos. Apenas para ponto de comparação, é relevante mencionar a tecnologia CDMA que é usada nos Estados Unidos da América. Nesta rede, para o utilizador mudar de telemóvel sem trocar de número, é necessário dirigir-se à operadora, pois não usa cartão SIM.

Ainda dentro da geração 2.5G, é imprescindível relembrar que foram sendo feitos progressos ao longo do tempo, nomeadamente o EDGE (Enhanced Data rates for GSM Evolution) e o HSCSD (High-speed Circuit-Switched Data), que permitiam velocidades até 236kbps e 57.6kbps respectivamente.

Devido ao desempenho que o EDGE tinha, na altura, foi possível assistir a um *streaming*, ou seja transmissão de dados contínuos, através de um acesso móvel, o que foi uma grande evolução na altura [42] [43] [44].

Depois surgiu a terceira geração de sistemas de comunicações móveis, 3G, que foi apelidada de UMTS (Universal Mobile Telecommunications System), e, com esta tecnologia, veio reforçar a possibilidade de aceder à internet em dispositivos móveis, devido à velocidade que agora o 3G consegue alcançar, pois atinge velocidades na ordem dos 2 Mbps. Para além da tecnologia 3G poder ser usada nos telemóveis, com o aumento da velocidade, começou-se a usar esta tecnologia num género de pen/ modem. Mais recentemente encaminhadores com autonomia própria onde se insere um cartão sim, podendo assim fornecer internet em qualquer parte do mundo, desde que tenha cobertura, podendo, assim, servir um vasto grupo de pessoas, a nível de acesso à internet que se encontrem perto do encaminhador sendo a ligação feita também através de usb. O acesso à internet seria então com qualidade similar às ligações fixas, podendo assim conseguir aproveitar todos os recursos que a internet proporciona, tais como *streaming* de vídeo, aplicações de áudio, entre outros. Esta tecnologia recorre ao uso do W-CDMA (Wideband Code Division Multiple Access), assim como a HSPA (High Speed Packet Access) que consiste na junção de dois protocolos de redes móveis, o HSDPA (High Speed Downlink Packet Access) e o HSUPA (High Speed Uplink Packet Access), que permite uma velocidade máxima de download de 14,4 Mbps e para upload 5.76 Mbps respectivamente. Houve uma evolução do presente HSPA, criando assim o HSPA+ com velocidades que ascendem os 168 Mbps de download e 22 Mbps de upload. Para alcançar estes resultados, recorreu-se à técnica de modulação de ordem superior (64QAM) e MIMO (Multiple-Input, Multiple-Output). No entanto, os operadores optam sempre por limitar a velocidade, de forma a tentar assegurar que conseguem fornecer o serviço de forma estável, a todos os seus clientes [44] [45].

Com o decorrer do tempo, e com o avanço da tecnologia surgiu a quarta geração de sistemas de comunicações móveis com o nome de LTE (Long Term Evolution). Possui uma velocidade máxima teórica de 300 Mbps de *download* e 75 Mbps de *upload*. Para o *download* é utilizada a tecnologia OFDMA (Orthogonal Frequency Division Multiple Access, em português, Acesso Múltiplo por Divisão Ortogonal da Frequência) e em relação ao upload o que se utiliza é SC-FDMA (Single

Carrier Frequency Division Multiple Access, em português, acesso múltiplo por divisão da frequência com Portadora Única). Tanto a OFDMA como a FDMA são semelhantes e o seu uso permite reduzir o consumo dos aparelhos a elas ligados. Tendo em conta que a tecnologia LTE é um sistema baseado em IP, sendo uma manipulação e simplificação da arquitectura do 3G, o que resultou numa diminuição de latência, portanto, devido a isso, as redes LTE e as redes 2g e 3g não são compatíveis, além de trabalharem num espectro de frequência diferente. Isso implica que montar uma rede LTE, tenha de ser construída desde o zero. Isto é o que tem atrasado a implementação da rede LTE [45] [44].

		3G	4G	Melhoria 4G
Speedtest (média de 3 testes)	Download (kbps)	6456	18073	2.8x
	Upload (kbps)	1969	32979	16.7x
	Ping (ms)	106	58	1.8x
Navegação Web (média de 6 testes)	Carregamento de páginas	10.3s	5.8s	1.8x
YouTube	Iniciar reprodução Vídeo HD	9.9s	2.7s	3.7x
Play Store	Instalar Google Earth	17.6s	7.5s	2.3x
MÉDIA FINAL				3.5x

Tabela 2.3 - Tabela comparativa de *ping* entre tecnologia 3G e 4G [46]

No entanto, actualmente, apenas se tem vantagem em utilizar o 4G para dados, tal como se comprova na tabela 2.3, pois o VoLTE (Voice over LTE), não está implementado em Portugal, sendo necessário para o telemóvel comutar para uma rede 3G, por exemplo, para conseguir efectuar e receber a chamada.

Apesar de já estar implementado em vários países, o LTE-Advanced, ainda não está em funcionamento em Portugal, no entanto, tem algumas características bastantes satisfatórias tais como uma taxa de *download* de 3 Gbps e de *upload* de 1.5 Gbps, além de aumentar o número de possíveis utilizadores em simultâneo [47].

Existe ainda uma outra tecnologia apelidada de 5G, que já está em teste há algum tempo, pela Ericsson e espera-se que em 2020 já possa começar a ser implementada comercialmente. Os testes demonstram que a tecnologia pode atingir os 20 Gbps de *download*, diminuir a latência e reduzir o consumo, levando a que aumente a autonomia do aparelho [48] [49].

Saliente-se que é possível o telemóvel deixar de apanhar uma tecnologia que use W-CDMA e apanhar apenas GSM, mesmo assim a chamada contínua, sem qualquer tipo de quebra ou interrupção.

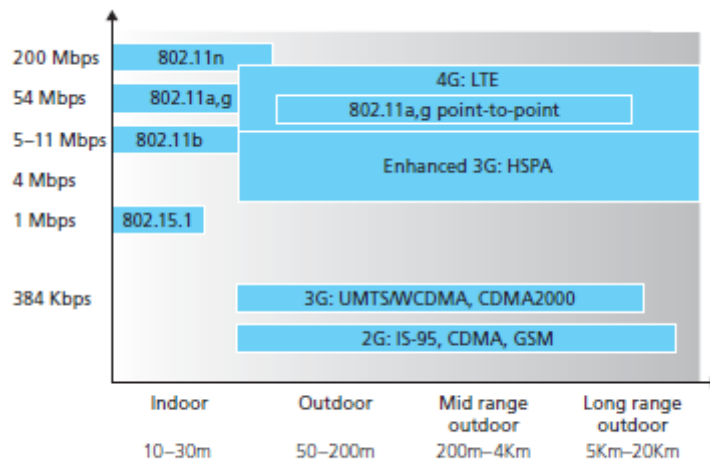


Figura 2.15 - Comparação de velocidade *versus* distância com várias tecnologias [50]

Na figura 2.15 consegue-se ter uma noção da evolução das redes móveis e das redes Wi-Fi, constatando que a velocidade de comunicação vai aumentando.

Uma grande vantagem do GSM, devido a estar largamente implementado em todo o mundo, é que caso a pessoa viaje para um país, onde o GSM trabalhe numa frequência compatível com o seu telemóvel oriundo do país de origem, poderá usufruir da rede sem qualquer tipo de problema, sendo as frequências 900MHz, 1800MHz e 1900 MHz as mais comuns.

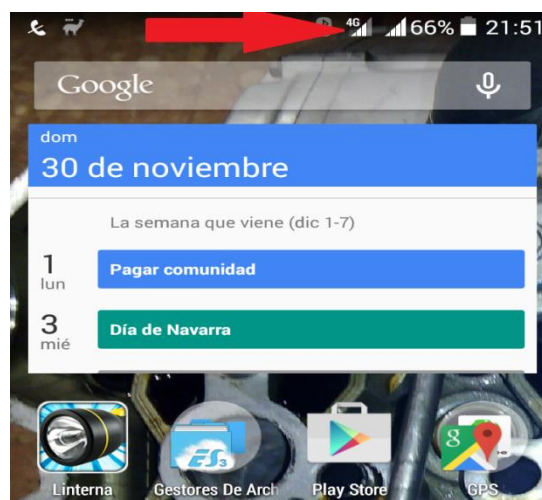


Figura 2.16 - Indicador de redes móveis [51]

No dia a dia, já se utilizou estas tecnologias, mas provavelmente sem se ter a percepção disso, pois por exemplo, basta olhar para a figura 2.16 e repara-se que no sítio onde mostra a quantidade de sinal de rede móvel que se consegue apanhar, aparece por cima umas letras. Neste caso é 4G que significa que está a apanhar rede LTE. Pode também aparecer E, o que significa que está a apanhar rede EDGE, G significa GPRS, H significa HSDPA, H+ significa HSDPA+ que é uma evolução do anterior, se aparecer 3G poderá estar a apanhar UMTS/W-CDMA e se aparecer 3.5G é HSDPA [51].



Figura 2.17 - Indicações numa caixa de um *smartphone*

Na figura 2.17, pode-se observar as frequências de trabalho que, neste caso o Vodafone Smart Ultra 6 utiliza e pode-se verificar, tal como indicado anteriormente, que este smartphone trabalha nas frequências 900/1800/1900 Mhz na rede GSM, sendo estas efectivamente as mais usadas em Portugal. Mais uma vez, tal como mencionado em cima, caso se desloque para outro país, apenas é necessário ter o cuidado de confirmar que as frequências usadas são as mesmas e assim não existirá qualquer tipo de constrangimento. Pode-se ainda verificar que, em termos de rede 3G (UMTS), usa as frequências 2100/900 Mhz e 4G (LTE), as frequências 2600/1800/800/900 Mhz, sendo estas as mais frequentes.

Outro tipo de característica que está presente, desde o GSM, é o facto de todos os aparelhos terem um IMEI associado (International Mobile Equipment Identity). Um IMEI é uma sequência numérica única para cada aparelho, definida pelo fabricante, onde as oito iniciais possuem informação do modelo e fabricante, os seis seguintes correspondem ao número de série e o ultimo número é para verificar se o IMEI é válido. É uma grande vantagem guardar-se o IMEI num local seguro, pois em caso de roubo ou perda, poderá ser facilmente bloqueado remotamente e, assim, impedindo de ser utilizado [42].

Existe ainda uma tecnologia chamada CDMA (Code Division Multiple Access), que foi utilizada em 2001 pela Zapp. Essa tecnologia é bastante diferente do AMPS, D-AMPS e do GSM, pois em vez de dividir a faixa de frequências permitidas em canais, o CDMA permite que cada estação transmita usando o espectro disponível, para tal. As diferentes transmissões em simultâneo são separadas recorrendo a codificação para não existir interferências [41].

Princípios básicos de redes de comunicação Móvel no GSM/UMTS

É necessário fazer com que se utilizem pequenas potências, existir a reutilização das frequências, recorrendo a pequenos grupos de células, seguimento dinâmico do movimento do utilizador, modificando os canais de rádio que lhe são atribuídos, que é o processo usado nas redes sem fio para tratar da transição de um telefone de uma célula para outra, de forma transparente para o utilizador, o que também se pode chamar de *handoff/handover*.

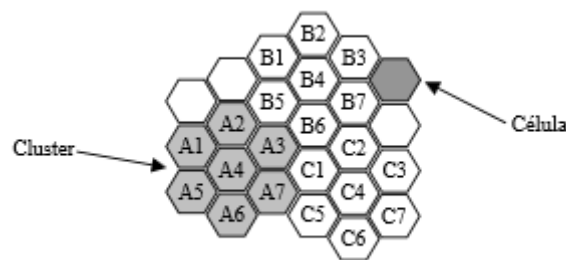


Figura 2.18 - Exemplo de um *cluster* constituído por 7 células [44]

Cada transmissor cobre uma área de forma hexagonal denominada célula, apesar de que na prática, a área não será exactamente hexagonal, devido a irregularidades da geografia do terreno, das condições de propagação e, portanto, consistirá numa forma irregular.

Tem-se como objectivo poder reutilizar os canais/frequências em diferentes regiões geográficas. Para isso, as células são agrupadas em grupos formando assim os *cluster's*, tal como na figura 2.18. Este tipo de estrutura repete-se ao longo da área geográfica, que é coberta pelo transmissor, fazendo com que exista a mesma separação espacial entre duas células que usem o mesmo canal, tal como se pode verificar na figura 2.18, no caso do A1, B1 e C1. Esta separação, juntamente com as pequenas potências utilizadas nos transmissores, faz com que a interferência entre canais reutilizados tenha valores aceitáveis.

Tendo em conta que o número de pessoas, que estão presentes nas redes móveis, está constantemente a aumentar numa dada área, é necessário pensar nalguma solução. Essa solução para resolver possíveis problemas de congestionamento da rede, passa por dividir as células dessa área em células mais pequenas, recorrendo a transmissores de menor potência, adaptando as células actuais é possível acompanhar o crescimento gradual e adequado do sistema [50] [44].

2.9 Sistemas Operativos

2.9.1 Android

O Android foi criado em 2003, sendo que a ideia principal era criar um sistema operativo avançado para camaras digitais. No entanto, quando foi lançado não tinha mercado suficiente e direccionaram o sistema operativo para telemóveis de forma a rivalizar com os sistemas da altura, Symbian e Microsoft Windows mobile. Em 2005, foi adquirido pela Google mantendo a equipa original e seguindo o caminho de ser um sistema operativo para aparelhos móveis, que era baseado no kernel do Linux, contendo uma interface para o utilizador que é baseada numa manipulação directa e simples. É feito essencialmente para dispositivos tácteis, como por exemplo *smartphones* ou *tablets* [52].

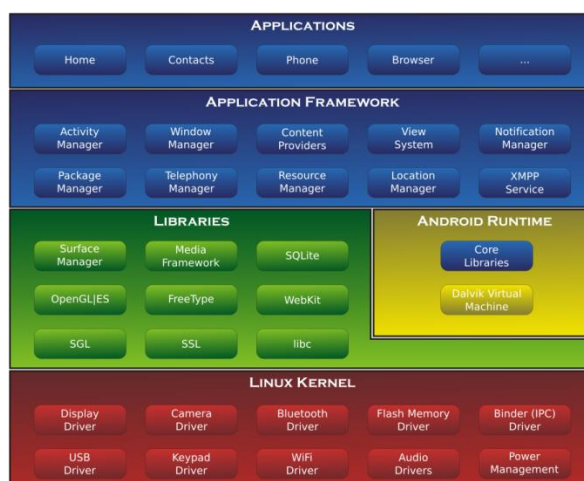


Figura 2.19 - Arquitectura do sistema Android [53]

Na figura 2.19, pode-se verificar que existe um baixo nível, onde existe todo o tipo de drivers e sistema de gerenciamento necessário para o Android funcionar. Sem esta camada, o Android não conseguia ler os dados que provem do *hardware*, controlar processos, gerenciar memória, entre outros. Tal como o nome indica, Linux kernel, o núcleo do sistema Android é derivado do kernel do linux. A camada de biblioteca, normalmente possui bibliotecas em c/c++ que são utilizadas

pelo sistema que lhe permite renderizações 3D, funções de acesso a base de dados SQLite, entre outras. Na camada do Android runtime é criada uma máquina virtual Dalvik, cada vez que uma aplicação é iniciada. Esta máquina virtual permite melhor desempenho, melhor integração com o *hardware* que vai saindo e foi projectada para poderem ser utilizadas várias em simultâneo. É devido a existir uma camada Application Framework que se consegue fazer aplicações posteriormente, pois adquire-se acesso a todas as funções que se tem disponível no Android, podendo assim ser usadas na aplicação e assim consegue-se interagir com o smartphone. A camada de aplicações é onde se encontram todos os programas que estão a correr, tais como, cliente de sms e mms, cliente de email, navegador, mapas calculadora, etc [54] [55].

O Android é agora também usado em aparelhos que se podem ligar a uma televisão, podendo assim usufruir de toda a versatilidade que o Android permite, à distância de um teclado Bluetooth por exemplo. O código fonte do Android é open source, mas por norma as marcas que o usam, colocam algum software proprietário, para incluir algumas funções que não vêm de origem.

O Android começou a ser usado em smartphones em 2008, com a primeira versão comercial designada de Android 1.0. Desde 2009 que as versões de Android são desenvolvidos mediante um nome de código passando por Cupcake, Donut, Eclair, Froyo, Gingerbread, Honeycomb, Ice Cream Sandwich, Jelly Bean, KitKat e actualmente Lollipop (Android 5.0), à medida que iam saindo novas versões eram adicionadas mais funcionalidades, segurança melhorada, design diferente ou simplesmente corrigir erros existentes em versões anteriores [56] [52].

A sua grande vantagem é ser *software open source*, pois isso significa que qualquer pessoa pode modificar o *software* original. É possível, por exemplo, fazer com que o processador corra a uma frequência mais elevada para ter mais desempenho ou diminuir a frequência do processador e assim fazendo com que a bateria dure mais. Além de que através do Android SDK (Software Development Kit) é possível criar aplicações usando uma linguagem muito semelhante ao java [57].

2.9.2 IOS

O IOS, também chamado de Iphone OS, foi inicialmente usado em 2007 quando o primeiro Iphone foi lançado. Foi o termo usado para descrever o sistema operativo, que iria funcionar nos Iphones, sendo que deriva do termo “OS X”, que é como a Apple descreve o sistema operativo que usa nos Macintosh, onde o X significa a versão do sistema. A plataforma IOS é um sistema operativo para dispositivos móveis que funciona como um sistema de computador, mas que é usado em iphones, ipods e ipads. Foi feito para ser um sistema pequeno, rápido e que usasse pouca energia. Foi ainda acrescentada uma interface touch que possibilita a fácil utilização em vez de usar um rato e um teclado, como foi usado ao longo dos anos [58] [59].

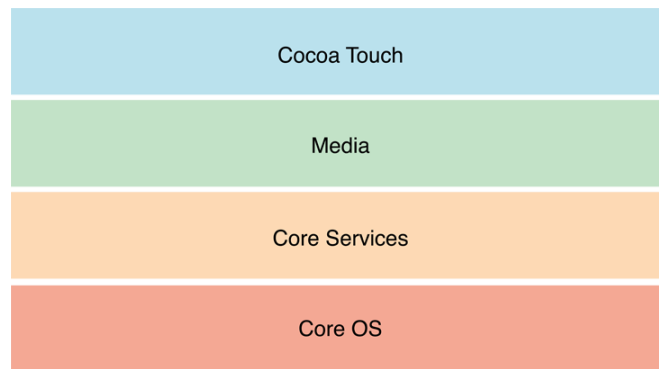


Figura 2.20 - Camadas do sistema operativo IOS

Tal como se verifica na figura 2.20, o IOS tem várias camadas:

- A camada Cocoa Touch contém as *frameworks* (código/funções que podem ser usados na aplicação) necessárias para criar aplicações para o IOS. A *framework* também define a aparência que a aplicação irá apresentar, tal como também providencia a estrutura básica desta. Também cria o suporte para as tecnologias que o telemóvel tem, tal como *multi-tasking*, *touch*, entre outros.
- A camada Media contém os gráficos, som e tecnologias de vídeos que se usa para implementar as experiências multimédia, na aplicação que se está a desenvolver.
- A camada Core Services contém os serviços fundamentais do sistema para a aplicação funcionar. Inclui ainda a tecnologia que suporta a característica de localização, *Icloud*, rede de internet, entre outros.
- A camada Core OS contém as características de baixo nível, onde todas as outras tecnologias assentam. Mesmo que não se use essa tecnologia directamente na aplicação, é possível que esteja a ser usada por outro *framework* [60]

2.10 Evolução da internet IPv4/IPv6

O endereço IP (Internet Protocol) é um recurso crucial para o funcionamento da internet. Tal como um número de telefone, um endereço IP é associado a cada interface de rede, seja ele um encaminhador, computador, telefone, servidor, etc, com ligação à internet. Foi desenvolvido nos anos 70 com a finalidade de interligar computadores que se encontravam em diferentes redes. O nome internet foi o escolhido para designar o protocolo, posteriormente rede mundial de informação com o intuito de designar o conceito de inter redes, que significa conexão entre redes. No início, o protocolo apenas era utilizado para fins militares, mas, rapidamente os computadores das universidades, das empresas e dos utilizadores particulares aderiram.

A versão mais comum é a IPv4 (Internet Protocol version 4), que foi lançado em 1978 e foi tornado padrão em 1981. IPv4 é a quarta revisão do desenvolvimento do protocolo de internet e

a primeira versão do protocolo a ser largamente implementada. Em 1980, o departamento de defesa dos Estados Unidos da América anuncia a migração do ARPANet para o IPv4, a 1 de Janeiro de 1983.

Tendo em conta que o IPv4 já está a ficar saturado, devido ao elevado número de dispositivos existentes, nos nossos dias, com ligação à internet, é necessário, portanto, fazer mudanças.

Actualmente já se está a pensar no IPv6, apesar de já ter sido estabelecido como o sucessor do IPv4 em 1995, pois o actual tem limitações. Por exemplo, quando um computador se liga à internet, é necessário que tenha um endereço IP único, tal como já foi mencionado acima, pois só assim se consegue identificar cada dispositivo que está ligado à internet. Para isso é necessário que uma entidade distribua os diferentes IP's, disponíveis às diferentes partes do mundo, onde, por sua vez, as entidades regionais distribuem pelos diferentes ISP (Internet Service Provider). Quando se contrata um serviço de internet, é necessário que o ISP tenha um endereço IP disponível para o cliente poder ser identificado na internet.

O problema que está a surgir é que os endereços IP estão a acabar, pois tendo em conta o formato do IPv4 x.x.x.x, onde x corresponde a 8 bits (octeto) significa que são 32 bits, sabendo que o octeto pode variar entre 0 e 255 (2^8). Então o número máximo de endereços disponíveis são 4.294.967.296, a este valor é necessário subtrair os endereços que são usados para classificação de redes, mascaras de redes, *broadcast*, etc.

Tendo em conta que a população mundial tem cerca de 7 bilhões, se for tido em conta que apenas metade das pessoas tem acesso à internet e algumas dessas pessoas possuem equipamentos moveis, que também necessitam de ligação à internet, rapidamente se atinge o limite do IPv4.

Para combater esta limitação, foi desenvolvida uma nova versão do IP, IPv6, que utiliza um número de 128 bits. O formato do IPv6 é do tipo xxxx.xxxx.xxxx.xxxx.xxxx.xxxx.xxxx.xxxx, ou seja, oito grupos de 4 dígitos hexadecimais, portanto, o número máximo de endereços IP é de 2 elevado a 128, o que significa que são $3,4$ com um expoente de 38.

Para além da vantagem acima mencionada relacionada com o número de IPs que a versão 6 possui, pode-se ainda referir que tem mais segurança, maior compatibilidade com aplicações de tempo-real, pois dará um melhor suporte ao tráfego (por exemplo videoconferência), dado que possui toda a informação de fluxo que tem de fazer/ percorrer nas suas especificações, possibilitando, assim, que os encaminhadores saibam a qual fluxo pertencem os pacotes

transmitidos. Incluirá também a norma “plug and play”, tal como o IPv4, mecanismo que facilita a ligação dos equipamentos dos utilizadores à rede [61] [62] [63].

Existe também o IPv5, mas que está atribuído a ST-II. O ST-II significa Stream Protocol version 2 e isto serviria para identificar de que tipo de pacote se tratava. Se fosse 4 seria um pacote normal, se fosse 5 seria um pacote de *stream* protocolo. Portanto, o número 5 não pode ser utilizado para designar a versão do protocolo IP que segue a versão 4, o que faz com que não haja um salto de versão. Simplesmente a versão 6 foi a sucessora da 4, pois a 5 está reservada para outro protocolo [63].

É necessário que o novo protocolo IP seja flexível de modo a permitir uma transição gradual. É imperativo a interoperabilidade entre as duas versões, pois, a quantidade de estruturas configuradas em IPv4 é elevada.

Para existir uma transição foi necessário criar um conjunto de mecanismos denominado de SIT (Simple Internet Transition). Este projecto foi feito de modo a simplificar a instalação e integração do IPv6 dos utilizadores, administradores de sistemas e operadores, tendo como objectivo a actualização progressiva e individual de servidores e encaminhadores, evitar que seja necessário actualizações e que a transição seja completada antes do esgotamento dos endereços IPv4. Os mecanismos propostos pelo SIT permitem que *hosts* IPv6 possam funcionar com *hosts* IPv4, até ao momento em que não existir mais nenhum endereço IPv4. Utilizando o SIT, garante-se de que a nova versão do protocolo IP não vai tornar a versão actual obsoleta, protegendo, assim, todo o investimento já feito no IPv4. No entanto, existem servidores que precisam de uma ligação limitada, por exemplo, uma impressora, nunca precisarão de ser actualizados para o IPv6. As técnicas que estão presentes no SIT são: os endereços IPv6 serão compatíveis com IPv4, criação de túneis de transporte de pacotes IPv6, através de topologias IPv4, juntar ao cabeçalho IPv4 o datagrama do IPv6 para transmissão de pacotes IPv6, através de topologias IPv4 e, por fim, uma camada de IP dupla, ou seja, uma implementação que permite suportar ambas as versões do IP, isto para servidores e encaminhadores.

Este processo vai ser lento e complexo, pois tem de se ter vários aspectos em conta. Tem de se ter bastante cuidado, pois caso se faça uma transição mal dimensionada ou demasiado rápida fará com que as pessoas/ os seus equipamentos não estejam com o protocolo de rede IPv6 instalado, logo não estão preparados e fará com que não se possam conectar à internet, o que iria causar bastante incómodo, pois actualmente faz-se tudo pela internet ou com recurso à internet.

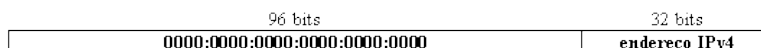


Figura 2.21 - Endereço IP6 compatível com IPv4 [64]

Tal como se pode verificar na figura 2.21, os túneis IPv6 em IPv4 usam uma sequência específica para mostrar que o IPv6 possui um IPv4, que é identificado por um prefixo de 96 bits a zero e transportando um endereço IPv4 nos 32 bits menos significativos.

Tal como foi mencionado, para permitir a comunicação entre nós apenas IPv6 e nós apenas IPv4, foi necessário criar uma camada dupla. Os nós que implementam estas camadas designam-se de nós IPv6/IPv4 e conseguem comunicar com as duas versões de IP. Foi necessário proceder a algumas alterações para permitir que seja utilizado aplicações sobre o IPv6, das quais destacou-se: as funções `getaddrname` e `getaddrinfo` foram alteradas de modo a permitirem efectuar mapeamentos de endereços de 128 bits, criação de novas funções, nomeadamente a `inet_ntop` e `inet_pton`, uma estrutura que possui a informação sobre o IPv6 junto com o número da porta do protocolo chamada de `sockaddr6_in` e uma estrutura que possui a informação sobre o IPv4 junto com o número da porta de protocolo chamada de `sockaddr_in`.

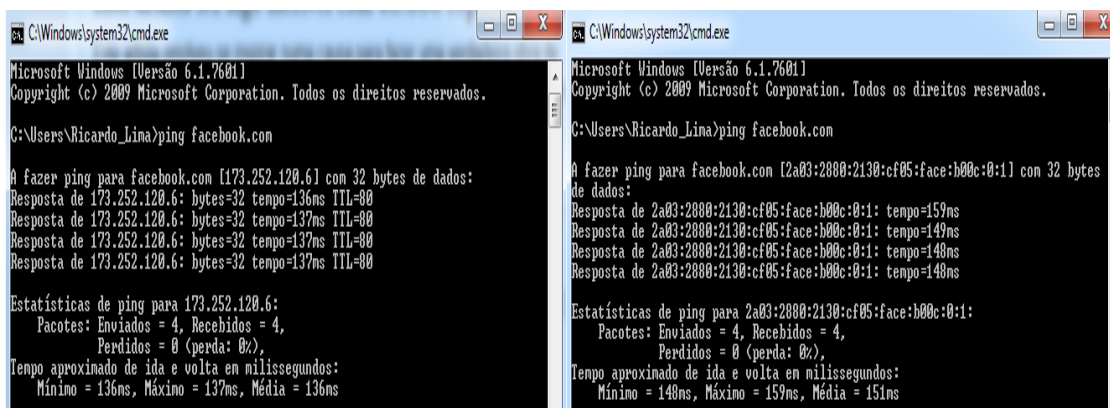


Figura 2.22 - À esquerda *ping* com IPv4 e à direita *ping* com IPv6

Na figura 2.22, consegue-se identificar uma sequência de números/letras que estão à frente do nome do site, irá ser explicado em seguida do que se trata.

Falando em IP's torna-se imperativo falar sobre DNS. As siglas DNS significam *Domain Name Service*, que consiste em ser capaz de converter IP's em nomes. Este serviço facilita bastante a vida das pessoas, pois imagine-se que se acede diariamente a 5 sites, seria necessário saber uma

quantidade de sequência de números enorme e se for em IPv6 ainda seria pior. Por exemplo neste momento o www.facebook.com tem o IP 173.252.120.6, o que significa que se for colocado o IP ou o endereço com o nome é igual, no entanto, sem dúvida que se torna mais cómodo colocar o nome. Se apenas se tiver activo na nossa placa de rede, o protocolo de rede IPv6, significa que se estará com um IP do tipo IPv6, então quando se fizer *ping* a um site, e esse site tenha também IPv6 configurado, irá fazer-se *ping* a um IP do tipo IPv6, tal como se pode verificar na figura 2.21.

O DNS tem de ser capaz de resolver registos do tipo A e AAAA que significa respectivamente endereços IPv4 e IPv6 [65].

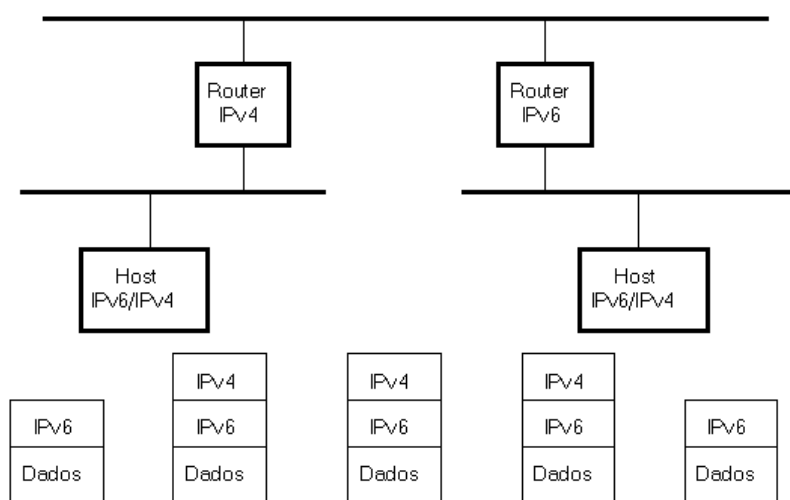


Figura 2.23 - Transmissão de pacotes através de túnel IPv6 em IPv4 [64]

Na foto 2.23, consegue-se verificar que é possível lidar com IPv4 e IPv6 em simultâneo, tendo em atenção que existem dois tipos de túneis, o configurado e o automático. Tem bastantes mecanismos na base que são comuns aos dois tipos, tais como: o nó de entrada faz o encapsulamento do pacote IPv6 num cabeçalho IPv4 e o nó de saída recebe o pacote encapsulado, retira o cabeçalho IPv4, torna-o em cabeçalho IPv6 e processa o pacote que daí resultou. “No túnel configurado o endereço do nó de saída do túnel é determinado com base em informação de configuração no nó onde se faz o encapsulamento. Este nó necessita de armazenar o endereço do final de cada túnel que nele se inicia. Quando um pacote IPv6 é transmitido através de um túnel, o endereço final configurado para esse túnel é usado como endereço destino do cabeçalho IPv4 que encapsula o pacote.

A escolha de quais os pacotes a enviar por cada túnel, é feita através de informação de encaminhamento do nó que vai encapsular esses pacotes.

No túnel automático, o endereço do nó de saída do túnel é determinado a partir do pacote que vai ser encapsulado. O endereço destino do pacote original tem de ser um endereço IPv6 compatível com IPv4, sendo o endereço do final do túnel a componente IPv4 do primeiro, i.e., os 32 bits menos significativos do endereço IPv6 compatível com IPv4.” [64].

2.11 Web services

Os *web services* permitem a integração entre sistemas e compatibilidade de aplicações, permitindo, assim, que novas aplicações possam interagir de forma fácil, com aquelas que já existem e sistemas, que sejam desenvolvidos em plataformas diferentes, sejam compatíveis.

Por exemplo, quando se está numa página de internet, onde se coloca o código postal e temos de imediato a localidade, há uma grande probabilidade de estar a usar um *web service*.

O XML (eXtensible Markup Language) é a tecnologia responsável pela característica de interoperabilidade, permitindo a comunicação entre aplicações desenvolvidas, em diferentes linguagens de programação.

Os *web services* devem ser definidos de forma consistente para poderem ser usados e existir uma interligação com outros serviços e aplicações. A WSDL (Web Services Description Language) é uma especificação W3C (World Wide Web Consortium) e esta fornece a linguagem para descrever as definições de *web services* com formato XML.

SOAP

O SOAP é um protocolo para chamar funções remotas através de RPC (Remote Procedure Call). RPC é uma técnica usada no desenvolvimento de aplicações tendo como finalidade permitir que um procedimento local se torne um procedimento alojado na internet, trabalhando em modo cliente-servidor. O SOAP é um padrão da indústria adoptado pelo W3C. A mensagem é transportada, através do protocolo http e os protocolos de segurança a usar são semelhantes aos presentes no capítulo de segurança.

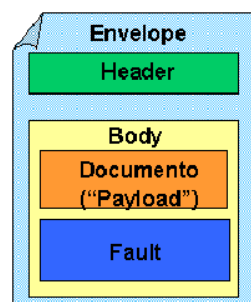


Figura 2.24 - Estrutura de uma mensagem SOAP [66]

A mensagem SOAP é constituída por cabeçalho e o corpo da mensagem, onde vem o documento e os erros, tal como está presente na figura 2.24.

REST

Neste modelo são definidos dois conceitos: cliente e servidor. Tal como no SOAP, o servidor disponibiliza um conjunto de serviços e o cliente usa o que necessita, tendo em conta o que tem à disposição.

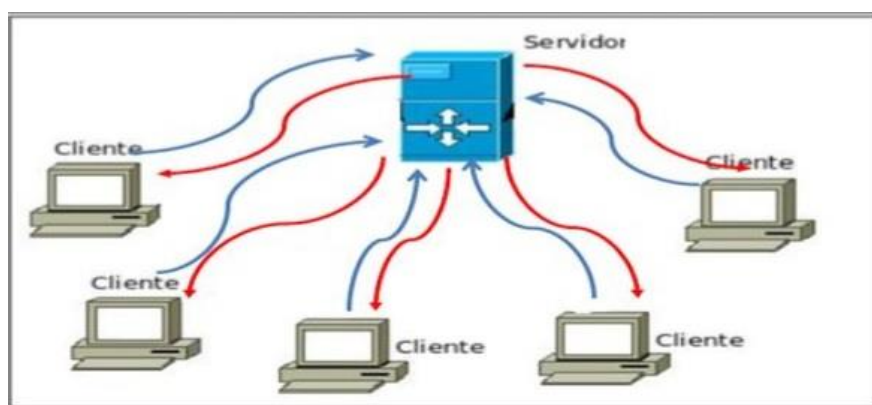


Figura 2.25 - Modelo cliente-servidor [67]

Clientes e servidor comunicam através da rede, onde o servidor está a fornecer serviços com os clientes. Observa-se este tipo de esquema regularmente no dia a dia, quando se vai ao email, quando se acede à internet, base de dados, com a configuração presente na figura 2.25.

O REST recorre ao padrão URI (Uniform Resource Identifier), podendo verificar isso no endereço do navegador, pois, aparecerá www.sitedaempresa.pt/aplicacao/funcao?parametro==x [67].

O REST é simples de entender e pode ser aplicado em praticamente qualquer cliente ou servidor com suporte a http/https. Muitas pessoas que o usam, dizem que como principais vantagens são a facilidade no desenvolvimento e aproveitamento da estrutura web existente.

No entanto, o REST é uma arquitectura e SOAP é um protocolo. O que fazem é combinar outras normas com o REST, nomeadamente os métodos GET, PUT, POST e DELETE, aumentando assim as suas funcionalidades [68].

2.12 Base de dados

Desde o início da história da Humanidade, que sempre houve uma necessidade do ser humano de registar e coleccionar dados. A prova disso é a descoberta do “The Dispilio Tablet” datada 5000

anos AC, ou até mesmo os hieróglifos encontrados no Egito. Ou seja, houve uma necessidade de criar uma representação de dados num espaço bidimensional [69].

Esta representação de dados, foi evoluindo, passando a uma escrita de linhas e colunas, originando assim, o que é denominado actualmente por tabelas. Para a manutenção destes dados, foi necessário criar bases de dados, pois estas tratam-se de um conjunto integrado de dados e/ ou elementos de informação que podem ser partilhados e utilizados concorrentemente por múltiplos utilizadores e/ ou programas, para múltiplos objectivos, e com diferentes perspectivas.

A base de dados é um sistema baseado em ficheiros, o que leva a alguns problemas:

- Redundância
- Inconsistência dos dados
- Maior esforço de introdução de dados
- Falta de flexibilidade
- Manutenção difícil
- Dificuldades de normalização (nomes, formatos, restrições)

Para fazer a gestão de uma base de dados é necessário um DBMS (Database Management System)/ SGBD (Sistema de Gestão de Base de Dados). Um DBMS é uma aplicação/ pacote de *software* que armazena e facilita a manutenção e acesso a grandes volumes de dados, sendo que foi desenhado para desenvolver e gerir bases de dados. Tem algumas vantagens das quais:

- Independência dos dados – não está exposto às aplicações a forma como os dados estão representados e armazenados
- Acesso e análise eficiente dos dados
- Integridade dos dados e segurança
- Administração uniforme e simplificada – é possível organizar a representação dos dados de forma a minimizar a redundância e o armazenamento e recolha de dados
- Acesso concorrente
- Robustez
- Tempo de desenvolvimento de aplicações reduzido
- Escalabilidade e eficiência no uso de recursos

No entanto, exige uma necessidade de especialização tecnológica e profissional e um maior investimento em *software* e *hardware* [70].

SQL vs NOSQL

Sem dúvida, que o tipo de base de dados mais conhecido e usado é o SQL (Structured Query Language). Este obriga a que sejam criadas tabelas e não podem conter dados que não sejam

relacionados, portanto, os dados entre si têm que ter alguma relação, o que resulta em serem chamadas também de base de dados relacionais. Já o NOSQL (Not Only SQL) são ficheiros em Json (JavaScript Object Notation), que podem ter a complexidade que for desejada, apenas é necessário organizar os ficheiros, com algum parâmetro. O formato Json é um formato leve de troca de informação. É fácil para os humanos lerem, escreverem e fácil para as máquinas dividirem e criarem mensagens [71].

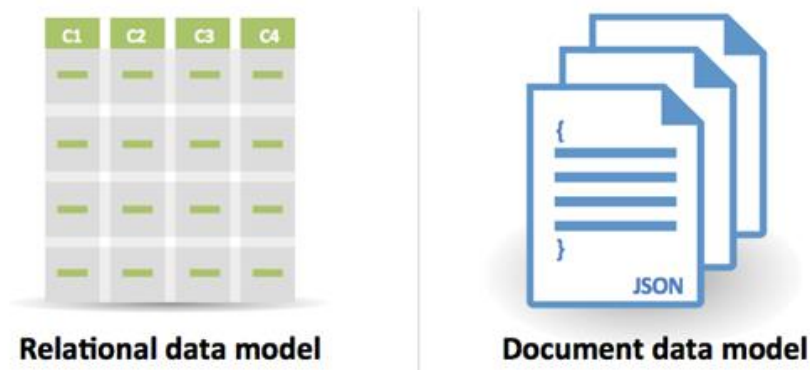


Figura 2.26 -Imagem demonstrativa de SQL e NOSQL [72]

No SQL é necessário existir uma estrutura com atributos definidos para guardar a informação, já no NOSQL possui-se a liberdade de usar a estrutura como for desejada, tal como é representada na figura 2.26. As consultas à base de dados SQL seguem sempre as mesmas normas, já o NOSQL tem várias formas de gerir os dados, dependendo de quem faz o programa. Ambas as soluções são facilmente escaláveis verticalmente, portanto é possível aumentar os recursos do sistema, no entanto, a solução baseada em NOSQL tem mais facilidade de escalar horizontalmente. As bases de dados SQL são mais consistentes e mais duráveis, contrariamente às bases de dados do tipo NOSQL que atingem maior performance e em ser mais facilmente escaláveis [73] [72].

A escolha entre SQL e NOSQL, está muito relacionado com o projecto que se tem em mãos e requer que seja analisado com cuidado.

3 Proposta

Neste capítulo apresentar-se-á uma proposta de solução, explicando a importância de cada sector. Para além da arquitectura da solução, onde se poderá constatar a comunicação entre as várias entidades. Ainda será possível verificar a escalabilidade do sistema.

3.1 Proposta de solução

Para se apresentar a proposta de solução é necessário pensar-se em dois locais distintos. Existirá um servidor pertencente à empresa que disponibilizará este produto e outro servidor que corresponderá a um suposto cliente. Com esta proposta um cliente conseguirá recolher a informação proveniente dos sensores, para isso recorrendo ao Gateway, o que vai possibilitar o envio para a internet e em seguida poderá verificar que se encontram disponíveis para consulta numa página na internet.

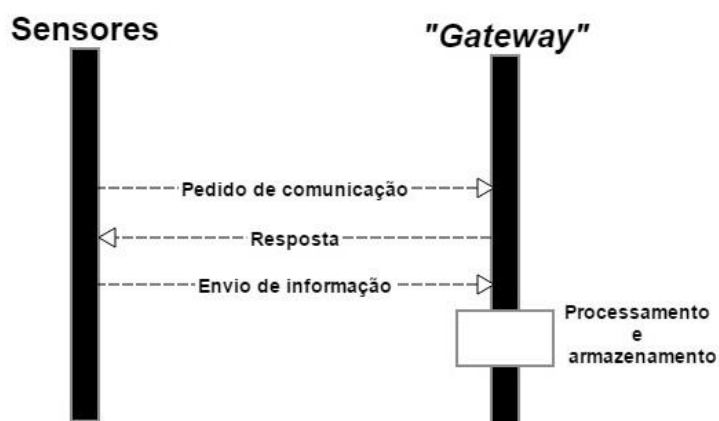


Figura 3.1 - Proposta de solução 1, presente na empresa

Na figura 3.1, verifica-se que existe duas entidades: sensores e *Gateway*. Estas duas entidades para comunicar vão recorrer a uma tecnologia que seja sem fios, para assim o *Gateway* poder recolher os dados, apenas deslocando-se perto dos sensores. Existirá um pedido de comunicação vindo dos sensores, seguido de uma resposta do *Gateway*, tendo em conta que será necessário determinadas condições para a ligação ser correctamente efectuada. Em seguida os sensores enviarão a informação de que dispõem e esta será processada e armazenada no *Gateway*.



Figura 3.2 - Proposta de solução 2, presente na empresa

No passo seguinte, na figura 3.2, o *Gateway* enviará a informação previamente guardada, para um programa que criará uma pilha de informação. Existirá uma aplicação que irá possuir a capacidade de consumir a pilha de informação, podendo guardar e manipular a mesma. Em certas ocasiões poderá ser necessário existir uma confirmação de que existiu um envio de informação.

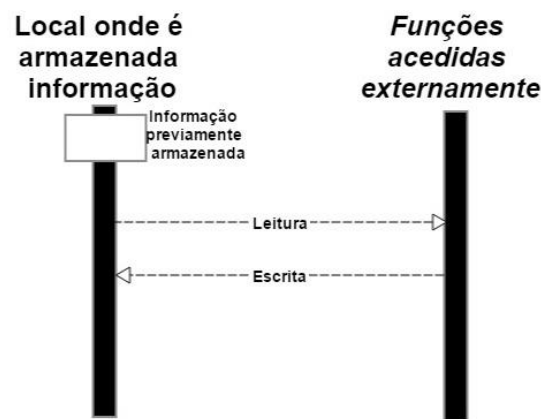


Figura 3.3 - Proposta de solução 3, presente na empresa

Para se poder ter acesso à informação que foi previamente armazenada, sem que para isso seja necessário dar acesso total, serão criadas funções que poderão ser acedidas externamente. Estas irão possuir a capacidade de ler e escrever informação, onde a informação está armazenada, tal como está representado na figura 3.3.

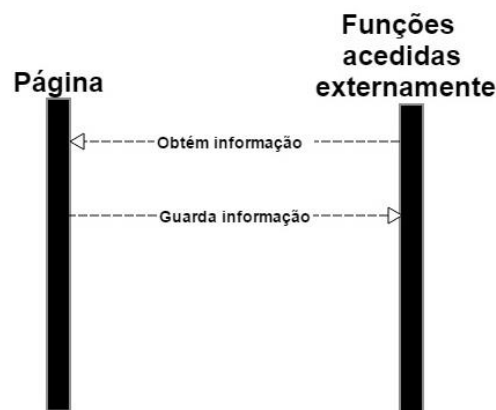


Figura 3.4 - Proposta de solução 4, presente na empresa e no cliente

Tendo em conta a figura 3.4, existirá uma página que para ser visível na internet é necessário que seja alojada e exposta por algum mecanismo. Tanto no cliente, como na empresa irão ter uma página, mas com diferentes vertentes. Na empresa será possível visualizar informações referentes a configurações, já no cliente, este conseguirá finalmente visualizar os dados que foram recolhidos dos sensores.

3.2 Arquitectura da solução proposta

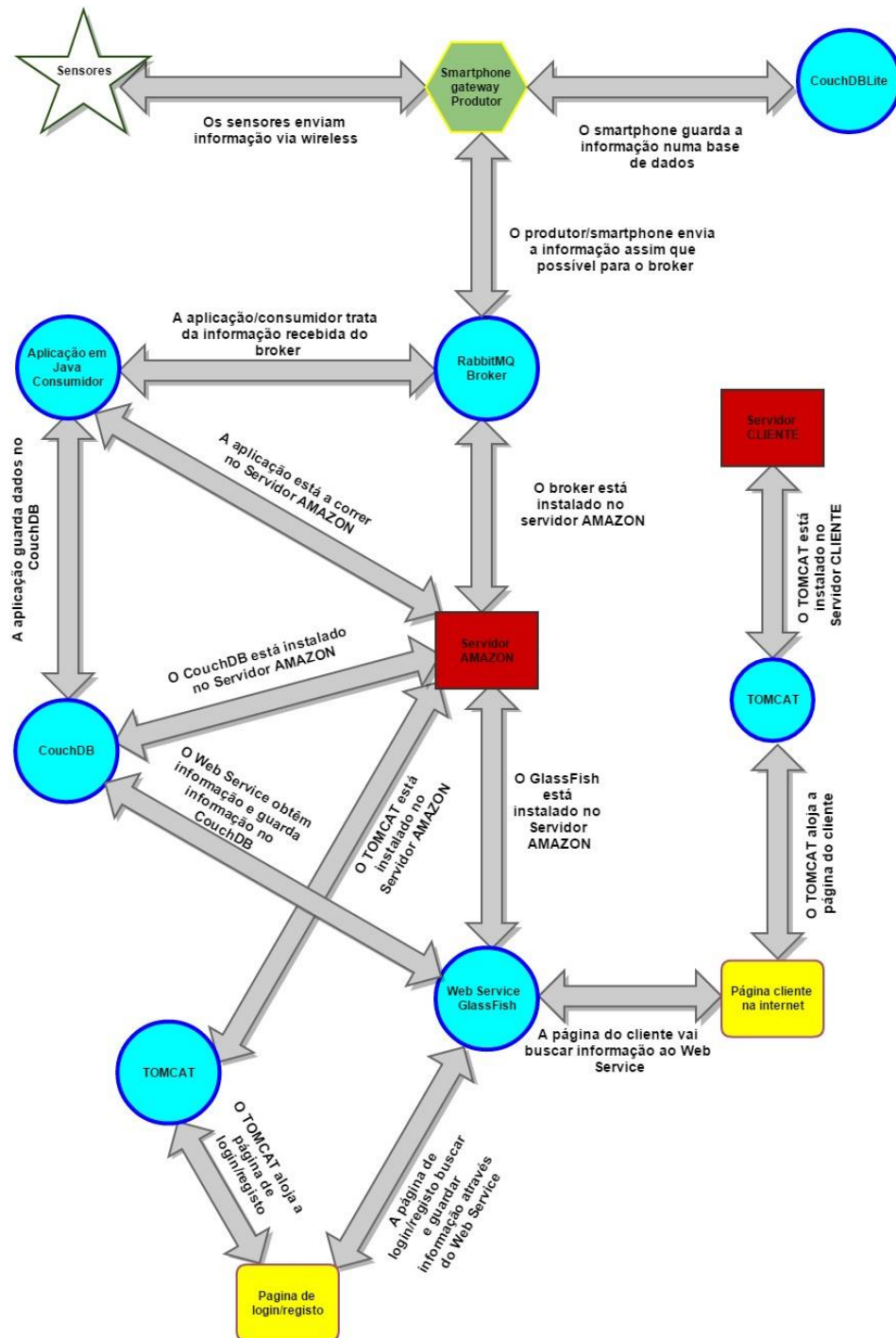


Figura 3.5 -Arquitectura do projecto



Figura 3.6 - Arquitectura do projecto em blocos

Para melhor compreensão dividiu-se a arquitectura em blocos, com os respectivos atributos que lhe estão associados, tal como está representado na figura 3.6.

Tal como se verifica na figura 3.5, a solução proposta passa por utilizar um telemóvel como *gateway*. O telemóvel vai receber informação dos sensores através do Bluetooth, cuja ligação tem de seguir determinados parâmetros, pois necessita de se garantir que apenas telemóveis autorizados recolhem a informação.

Quando o *smartphone* se liga à internet guardará informação sobre 20 sensores. Assim que se liga, se o *smartphone* tiver a aplicação instalada este envia uma palavra passe que o identifica e fica assim a ligação estabelecida. É necessário esta palavra passe para apenas pessoas autorizadas se ligarem ao Arduino, recolhendo assim a informação dos sensores.

Assumindo que a ligação foi feita com sucesso e o *smartphone* já tem uma base de dados com informação de 20 sensores, este envia uma mensagem a avisar que vai enviar nova actualização da última data e em seguida envia uma mensagem composta por hora, dia, mês, ano e sensor. Caso o Arduino tenha informação anterior a essa data, vai apagar essa informação, pois já se tem a certeza que esta já se encontra na base de dados. Caso já não tenha datas inferiores ao que foi mandada pelo *smartphone*, significa que já lá passou outro *smartphone* e já lhe deu a confirmação que estava na base de dados, apagando essa informação. O *smartphone* após ter verificado o que era necessário, manda um comando “Read Date”, fazendo com que o Arduino vá procurar o último ficheiro disponível para aquele sensor e envia.

O *smartphone* irá possuir duas bases de dados feitas em CouchDBLite. Uma irá ter informação dos sensores, nomeadamente a nível de nome, a quem pertence e a última actualização que está disponível na base de dado central. A outra irá conter os dados que foram lidos pelos sensores, juntamente com o seu nome e a quem pertence. Assim que recebe novos dados, estes são colocados na base de dados que se encontra no *smartphone*.

Antes de continuar é necessário explicar o conceito de Broker e todas as suas envolventes, nomeadamente produtor e consumidor. O Broker é um género de uma pilha de dados, ou seja,

vai recebendo os dados, pondo-os num género de fila de espera e existe um consumidor, que é uma aplicação em Java. Essa aplicação/consumidor vai retirando dados, a essa pilha de dados, guardando-os na CouchDB (base de dados central). De referir que o Broker mesmo que o servidor se desligue, ou deixe de possuir consumidor, guarda os dados em disco, assim que houver um consumidor disponível, o Broker irá fornecer os dados normalmente. O produtor como o nome indica, é o que produz as mensagens, e portanto, é o smartphone. De salientar que esse Broker está instalado no servidor AMAZON, tal como a aplicação em java e o CouchDB.

O utilizador poderá escolher quando envia os dados, podendo escolher entre ligação móvel ou Wi-Fi. Assim que tem possibilidade de enviar dados, são os dados mais antigos presentes na base de dados do telemóvel que irão seguir em direcção ao Broker. O telemóvel verifica sempre que tem internet, se a sua base de dados em termos de últimas actualizações, se encontra actualizada.

Neste momento os dados enviados pelo smartphone, já se encontram na CouchDB e portanto já poderão ser acedidos e visualizados.

Optou-se que o acesso à informação da base de dados CouchDB seria através de um *web service*, pois caso contrário, tornava-se necessário criar um utilizador individual para cada cliente, o que daria bastante trabalho e levaria a insegurança. Assim através de *web service*, este tem acesso total à CouchDB, mas só retira o que a função pede. O *web service* escolhido é o GlassFish e está instalado no servidor AMAZON.

Agora, é necessário que os utilizadores que utilizam este serviço de recolha de dados de sensores, sejam ele clientes particulares ou organizações, tenham acesso a verificar quantos sensores estão em seu nome, quantos estão activos, se há algum problema com algum deles, ou simplesmente apagar o seu utilizador dos registos, isto no caso particular. No caso de ser uma organização, poderá ver quantos utilizadores pertencem à mesma, poderá adicionar e remover utilizadores da sua organização e ver os sensores que lhes pertencem. Para apresentar essas informações, é necessário um servidor *web* (TomCat) que vai permitir interpretar as páginas e torná-las acessíveis na internet, também está instalado no servidor AMAZON. Estão disponíveis várias páginas na internet, das quais, *login* para utilizador comum e organização, para poderem ver as suas informações e páginas de registo para utilizador comum e organização, caso seja a primeira vez que vão utilizar o serviço.

Agora imaginando que um cliente genérico, seja ele particular ou organização, já se registou, já adicionou correctamente os seus sensores, já recolheu dados com o seu smartphone e já os enviou para a internet. O próximo passo é visualizá-los e para isso é necessário ter um servidor físico para poder instalar o Tomcat. Após já se ter o servidor e o Tomcat, só é necessário fazer a

página em php ou noutra linguagem que a pessoa se sinta confiante e irá ter acesso a todos os dados dos seus sensores, através do *web service*.

Escalabilidade da solução

```
Sep 05, 2015 4:15:26 PM org.lightcouch.CouchDbClient process
INFO: > PUT/mgse/29c44a3d2cb04c4ebc40033d0ff09526
Sep 05, 2015 4:15:26 PM org.lightcouch.CouchDbClient process
INFO: < Status: 201
Sep 05, 2015 4:15:26 PM org.lightcouch.CouchDbClient process
INFO: > PUT/mgse/230e4482c8494e0a89b466df32916517
Sep 05, 2015 4:15:26 PM org.lightcouch.CouchDbClient process
INFO: < Status: 201
Sep 05, 2015 4:15:26 PM org.lightcouch.CouchDbClient process
INFO: > GET/mgse/design/Organizations/_view/OrgaIds?include_docs=true
Sep 05, 2015 4:15:26 PM org.lightcouch.CouchDbClient process
INFO: < Status: 200
Waiting for clients...
{"type":"message_test","message":"ola"}
recebi a mensagem :ola
{"type":"message_test","message":"olannn"}
recebi a mensagem :olannn
{"type":"message_test","message":"olannn"}
recebi a mensagem :olannn
{"type":"message_test","message":"bacalhau"}
recebi a mensagem :bacalhau
{"type":"message_test","message":"bacalhau"}
recebi a mensagem :bacalhau
```

Figura 3.7 -Consumidor a ser executado no putty



Figura 3.8 - À esquerda um consumidor a correr no Eclipse e à direita o emulador do Android com a aplicação

Na figura 3.7 e 3.8 verifica-se que se tem dois consumidores/ duas aplicações em java, um a correr através do terminal putty e outro via eclipse, um produtor representado pelo emulador do Android e um Broker. Neste exemplo expandiu-se a arquitectura em termos de consumidores, lançando uma nova instância de um consumidor. Esta situação poderá ser vantajosa quando

existe uma pilha de dados com muita informação para ser tratada. Sendo dois consumidores a retirar informação da mesma, o tamanho da pilha de dados ficará mais controlado. No exemplo prático apresentado em cima, foi carregado 4 vezes em *message* para enviar a mensagem bacalhau e tendo chegado duas ao consumidor que está a correr no putty e outras duas ao que está a correr no eclipse, logo não houve perdas de mensagens.

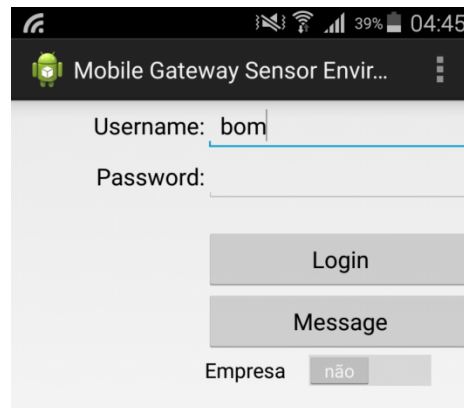


Figura 3.9 -Imagem do smartphone

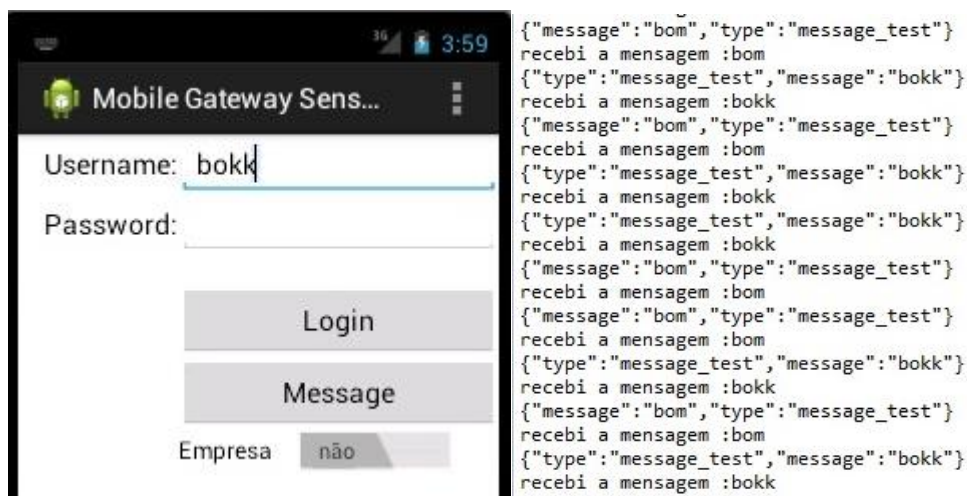


Figura 3.10 - À esquerda emulador do Android e à direita consumidor a correr no eclipse

Na figura 3.9 e 3.10 a arquitetura foi escalada em termos de produtores, portanto, um *smartphone* Android, um emulador Android, um Broker e um consumidor. Foi-se alternando o envio de mensagens entre bokk e bom, vindos respectivamente do emulador e do *smartphone*. Verificou-se que as mensagens vão chegando, sem qualquer tipo de perda. Ainda se pode ver que a mensagem em Json recebida “{“type” : “message_test” , “message” : “mensagem a receber”}”, pode mudar a sua ordenação mediante vem do emulador ou do *smartphone*, mas não tem

qualquer interferência, pois quando vai-se buscar a informação à mensagem do tipo Json, é através do campo e portanto não tem importância a ordem como chega.

Também é possível expandir a arquitectura do projecto em termos de Brokers, sendo para isso necessário iniciar logicamente duas instâncias de Brokers. Isto ajuda caso a fila de informação esteja a ficar muita grande, obrigando a que a informação proveniente dos *smartphones* seja repartida e assim aumentando o sistema a nível de performance.

Caso o sistema em termos de Broker, produtor e consumidor esteja a funcionar bem, mas o que esteja a atrasar o sistema é guardar ou aceder a dados na base de dados CouchDB, é possível replicar a base de dados, fazendo com que melhore os tempos, tanto de acesso, como de escrita, pois fica-se com duas bases de dados exactamente iguais, onde se uma tiver ocupada, o sistema reencaminha para a outra.

Ligação ao Broker por parte da aplicação em java

```
private final static String QUEUE_NAME = "MGSE_Message_Queue";
```

Figura 3.11 - Nome de canal do Broker

É necessário definir um nome para o canal de comunicação, tal como se verifica na figura 2.11. Cada aplicação que se ligue ao Broker necessita de saber o nome do canal ao qual se vai ligar. Caso contrário não terá acesso à informação.

```
ConnectionFactory factory = new ConnectionFactory();
factory.setHost(" . . .228");
factory.setUsername("ricardo");
factory.setPassword("");
factory.setVirtualHost("/MGSE");
factory.setAutomaticRecoveryEnabled(true);

Connection connection;
long delayTry = 100;
while (true) {
    try {
        connection = factory.newConnection();
    } catch (IOException e) {
        System.err.println("Failed to create connection. Retrying...");
        e.printStackTrace();

        try {
            Thread.sleep(delayTry);
        }
    }
}
```

Figura 3.12 - Ligação da aplicação em java ao Broker

Tal como se verifica na figura 3.12, a aplicação em java necessita de se ligar ao Broker, para assim poder retirar a informação que lá está guardada. No entanto necessita de algumas configurações, tal como o IP onde se encontra o Broker, o utilizador e a sua respectiva palavra passe e o VirtualHost. Todos estes parâmetros foram anteriormente configurados no próprio Broker e são necessários para se obter acesso ao canal.

4 Validação

Para se poder validar o conceito, foi necessário desenvolver aplicações, configurar aplicações. Ao longo deste capítulo poderá verificar-se os testes a que as aplicações foram submetidas de forma a comprovar o bom funcionamento das mesmas. Assim como as razões que foram tidas em conta, o que por sua vez levou às escolhas apresentadas. Tal como também uma cronologia das tarefas efectuadas.

4.1 Passos de implementação

- Criação de uma aplicação Android básica recorrendo às componentes `java.net.Socket` e `java.net.ServerSocket`, e também um servidor em Java. Foi testado localmente e realizou a comunicação com sucesso.
- Testou-se uma versão já com o RabbitMQ instalado localmente
- RabbitMQ instalado no servidor AMAZON
- Instalou-se o CouchDB
- Experimentou-se a Api ektorp, mas não funcionava correctamente. Optou-se pela lightCouch para comunicar com a base de dados CouchDB
- Implementou-se a função de login na aplicação Android
- Encriptação da palavra passe, tanto na base de dados, como na aplicação.
- Configurar base de dados na aplicação Android, a CouchDBLite
- Configuração do servidor CLIENTE, que consistiu em instalar a biblioteca php, para poder interpretar as páginas e o Tomcat, para fazer de servidor web. Inicialmente instalou-se Apache, mas depois optou-se por Tomcat
- Fez-se um *web service*, para isso instalando o GlassFish
- Foram realizados testes, já com páginas criadas em php, para aceder ao *web service* através do soapclient, primeiro localmente e depois num servidor
- Criação de um site simples, capaz de mostrar os sensores que estavam associados a um determinado utilizador

- Foi criada a página de registo de utilizadores e de login, onde poderá ver os seus sensores e apagar-se do sistema
- Criada página de registo de organizações e de login, onde poderá ver-se os utilizadores que estão associados à organização, podendo também apagar e registar utilizadores.
- Site e aplicação modificados de forma a trabalhar com organizações.

E muitos outros pormenores que foram sendo alinhados de acordo com o que era pretendido.

4.2 Análise das tecnologias

O facto de se ter feito estas escolhas não significa que sejam as melhores escolhas, significa sim, que foram as que melhor se adaptaram às necessidades na altura.

- Servidor – Foi escolhido um servidor da amazon, única e exclusivamente devido a ter-se um ano de testes grátis, o que era o tempo suficiente para testes. No entanto pode ser utilizado qualquer servidor desde que seja possível instalar Linux. A escolha de Linux para os servidores, foi uma escolha natural, pois por norma o Linux aguenta melhor os processos, tendo recursos mais escassos. Para demonstrar-se o conceito de cliente, procedeu-se à compra de um outro servidor, o mais barato que se conseguiu encontrar. Apenas com o requisito de conseguir correr Linux, acesso via SSH e acesso às portas de comunicação
- Android – As aplicações para testar este projecto poderiam ser efectuadas em Windows Mobile, IOS ou Android. A escolha recaiu sobre o Android, devido aos membros que elaboraram as dissertações possuírem telemóveis com o sistema operativo Android. Para além de que, tendo em conta a tabela 2.1 o Android era o sistema operativo com mais mercado.
- RabbitMQ – Tem a possibilidade do Broker ser clonado, evitando assim um problema, caso uma das máquinas falhe, pois as restantes continuam a funcionar e continuando assim a reter a informação. Pode existir vários Brokers a funcionar em simultâneo, mas formam apenas um Broker lógico, isto faz com que a performance aumente consideravelmente. Está disponível em várias linguagens de programação e tem uma larga comunidade a suportar e com bastante informação sobre esta tecnologia. Resumindo o RabbitMQ é um intermediário para mensagens. O RabbitMQ dá à aplicação criada, uma plataforma comum de receber e enviar mensagens, cujas mensagens permanecem num local seguro até serem recebidas. Apesar do RabbitMQ utilizar o AMQP

(Advanced Message Queuing protocol) e existir outros tal como o MQTT, activeMQ, FfMQ, HornetQ entre muitos outros. No entanto, devido a ser largamente comentado e ter uma boa documentação, optou-se por experimentar e cumpriu os requisitos. Requisitos estes que consistia em ser minimamente fácil de configurar, ter APIs para várias linguagens de programação e ter uma boa comunidade para ajudar a resolver possíveis erros. De referir que o RabbitMQ é usado no Mercadolivre e na aplicação Here [74] [75].

- Aplicação em java – Optou-se por desenvolver a aplicação em java, devido a já se ter alguma experiência em java, o que facilitou o desenvolvimento da aplicação.
- Base de dados – Aqui resume-se a uma grande diferença, NOSQL e SQL. Decidiu-se que iria fazer-se em NOSQL, devido a poder tratar os dados, sendo simples ficheiros em formato Json, o que na hora de se tratar os dados facilitaria, pois não se estaria limitado a campos de uma tabela. Existem situações em que por vezes uma tabela tem 10 campos, no entanto apenas 5 são preenchidos devido a não se enquadrar totalmente. Já nas bases de dados NOSQL, pode-se criar um ficheiro com os campos que se achar necessários. Dentro das base de dados NOSQL, existem várias hipóteses, do qual se destaca a CouchDB, MongoDB, Cassandra. Apesar do funcionamento ser bastante semelhante, a escolha recaiu sobre a CouchDB devido a ser feita pela apache, que já é bastante conhecida. Possui uma boa documentação, o que ajudou na implementação. Tem uma característica bastante importante, tem uma interface gráfica, chamada Futon, o que ajuda bastante no desenvolvimento. Esta permite aceder através do navegador, podendo criar ou destruir base de dados, ver e editar ficheiros, criar e correr *views*. Para aceder à CouchDB poderia ser via URL ou recorrendo a uma API. A escolha para o acesso a esta foi através duma API, que recaiu sobre o LightCouch, devido a ter sido actualizado há pouco tempo, o que é extremamente importante. No entanto foi testado com outros APIs disponíveis no site oficial do CouchDB, mas sem efeito. Na base de dados no telemóvel, optou-se por utilizar a CouchDBLite, devido a pertencer à CouchDB. Isto poderia trazer vantagens de integração em termos de replicar base de dados. Caso o utilizador quisesse, por exemplo, replicar os seus dados na íntegra na base de dados do seu telemóvel, para assim ter maior controlo e certificar-se com que os dados chegavam ao seu destino. No entanto, tal funcionalidade não foi implementada
- Tomcat - Foi escolhido o Tomcat para fazer de *web server*, mas podia ter sido outro qualquer web server. Para o nosso caso não havia grande diferença ser o apache, Tomcat (também pertence à apache mas escrito em linguagem diferentes), JBoss. Seria necessário instalar à parte o extra PHP, para conseguir lidar com as páginas escritas em

php. Foi escolhido o php, devido a ser uma linguagem de simples aprendizagem, devido a já se ter uma ideia geral do funcionamento e ter uma grande comunidade. No entanto, poderia ser utilizada uma qualquer linguagem web, desde que fosse compatível com os programas que foram usados no projecto

- Web service – Inicialmente foi necessário escolher que linguagem usar para fazer o *web service*. Optou-se por java, devido a já se estar familiarizado com tal. De referir que não importa em que linguagem é feita o *web service*, pois todos vão comunicar da mesma forma (tem de utilizar a mesma forma de comunicar, seja ela SOAP ou REST). A escolha recaiu sobre o protocolo Soap, devido a ser considerada a interface web mais antiga, o que leva a que a sua segurança, a sua comunicação, etc, já estejam bastante evoluídas e testadas. Além de possuírem uma vasta comunidade, o que ajuda na resolução de erros. No entanto, ambas as tecnologias fariam o trabalho pretendido sendo bastante fiáveis no mercado actual, mas devido à pesquisa que foi efectuada e seguindo este tutorial <https://netbeans.org/kb/docs/websvc/jax-ws.html>, reparou-se que este seria de fácil implementação e faria o pretendido. Optou-se pelo GlassFish, devido a ser largamente utilizado, ter uma boa comunidade e ser usado no tutorial o que ajudou bastante na configuração. Era possível instalar um extra ao Tomcat chamado de axis pertencente também à Apache, ficando assim com a possibilidade de fazer *web services*. No entanto, optou-se por separar os programas para evitar possíveis problemas. Poderia ser outro programa a fazer o trabalho de *web service*, no entanto experimentou-se este e verificou-se que era de fácil implementação e cumpria os requisitos que se pretendia, ser simples e eficaz. Para conseguir aceder a estas funções do *web service*, era necessário uma biblioteca específica. Inicialmente começou-se por utilizar a SoapClient passando os parâmetros por cabeçalho, no entanto este não conseguia lidar muito bem quando era necessário passar parâmetros por função, passando a utilizar-se a biblioteca NuSoap.
- Wi-Fi ou Redes móveis – A forma de envio dos dados para a internet, recaiu sobre o Wi-Fi e o Bluetooth, pois tendo em conta que o público alvo que poderá usufruir deste conceito, necessita obrigatoriamente ter em sua posse um *smartphone* para poder instalar a aplicação. Estes equipamentos por norma possuem todos Wi-Fi e redes móveis, portanto abrangerá um grande número de potenciais utilizadores.

4.3 Resultados de simulação

Nesta secção irá mostrar-se os resultados que são obtidos na prática, através de programas, páginas, etc. que foram efectuadas para demonstrar este conceito.

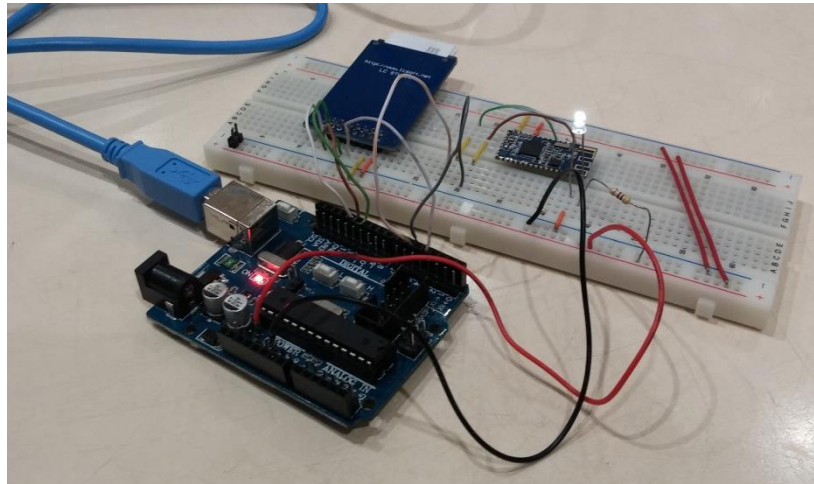


Figura 4.1 - Montagem do Arduino com Bluetooth e cartão SD

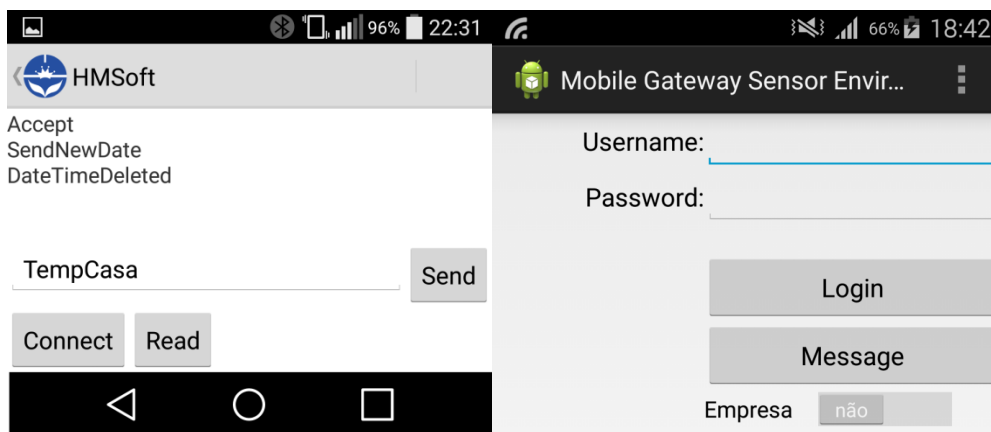


Figura 4.2 – Referente às duas aplicações que foram usadas para testes

À esquerda da figura 4.2 a aplicação HMSOft foi usada para testar a comunicação com Arduino com Bluetooth acoplado, figura 4.1. Na figura 4.2 à direita, a aplicação Mobile Gateway Sensor Environment (MGSE) que irá servir para testar o login e poderá também enviar-se mensagens para o servidor a simular informação vinda do sensor. Estas duas aplicações, caso fossem juntas, simulavam o comportamento da recepção dos dados, enviados através do Bluetooth, sendo que

essa informação estaria presente no cartão do Arduino. Mas para a realização de testes optou-se por estarem separadas.



Figura 4.3 - O HMSoft a receber informação do cartão simulando as medições dos sensores em formato Json

Portanto iniciar-se-ia carregando no botão Connect, presente na figura 4.3, fazendo todas as verificações necessárias. Após estas estarem concluídas, seria necessário carregar no botão Read e ter-se-ia dados vindos do cartão SD. O botão Send serviu para fazer os testes iniciais dos comandos que o Bluetooth necessitava para aceitar a ligação, sendo que agora a ligação é feita de forma automática. Agora já se tem dados em formato Json na aplicação HMSoft.

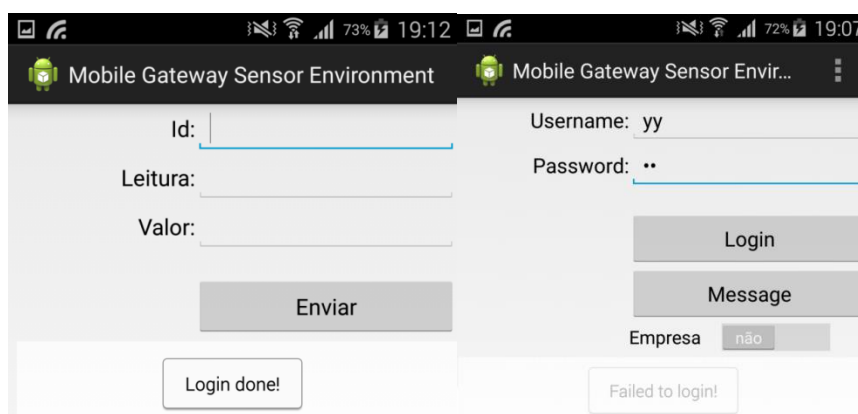


Figura 4.4 - Vários estados de login

Passando para a MGSE, é possível verificar que o sistema de login funciona. Quando se erra seja na palavra passe ou no utilizador, apresenta um erro indicando que falhou o login. Caso os dados colocados estejam correctos, é apresentado uma mensagem de login feito e expõe uma nova página, tal como se verifica na figura 4.4.

```
dataManager.saveData("{\"date\": \"12/12/2015 23:43:01\", \"sensorId\": \"3874\",  
  \"data\": [{\"date\": \"12/12/2015 23:43:01\", \"max\": 23.5, \"avg\": 2, \"min\": 23.2}, {\"date\": \"
```

Figura 4.5 - Função para guardar dados na CouchDBLite no smartphone

```
public void saveData(String data) {  
    Map<String, Object> docContent = new HashMap<String, Object>();  
  
    JSONObject jObs;  
    try {  
        jObs = new JSONObject(data);  
        docContent.put("date", jObs.getString("date"));  
        docContent.put("sensorId", jObs.getString("sensorId"));  
        docContent.put("signature", jObs.getString("signature"));  
        docContent.put("data", jObs.getString("data"));  
    } catch (JSONException e) {  
        // TODO Auto-generated catch block  
        e.printStackTrace();  
        return;  
    }  
  
    Document doc = db.createDocument();  
    try {  
        doc.putProperties(docContent);  
    } catch (CouchbaseLiteException e) {  
        // TODO Auto-generated catch block  
        e.printStackTrace();  
    }  
}
```

Figura 4.6 - Programação implementada em Android para lidar com o pedido para guardar dados na CouchDBLite

Para testar o armazenamento de supostos dados vindos de sensores criou-se uma mensagem em Json com informação, como se consegue verificar na figura 4.5. Na figura 4.6 tem-se a função previamente chamada, contendo todos os passos necessários para criar um ficheiro com a informação passada por parâmetro e assim ficar armazenada até ser tratada.

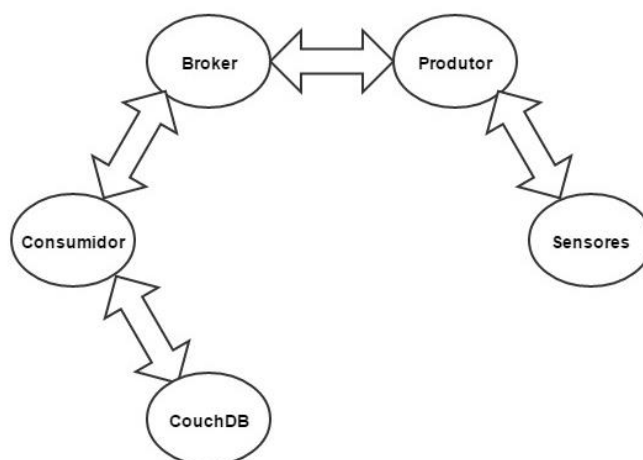


Figura 4.7 - Esquema de *login* e envio de mensagens

O *login* e envio de mensagens assentam na estrutura apresentada na figura 4.7. Para efectuar um login é necessário fazer um pedido de *login* do produtor para o Broker. O Broker reencaminha esse pedido para o consumidor. O consumidor vai verificar os dados ao CouchDB, verificando se os dados são correctos ou não. O consumidor reencaminha a resposta para o Broker e em seguida este envia para o produtor, sabendo assim se o *login* foi feito com sucesso ou não.

No caso de ser envio de mensagem, o produtor recebe os dados do sensor. O produtor envia os dados para o Broker, este reencaminha para o consumidor e este manda guardar a informação na CouchDB.

```
} else if (jObs.getString("type").equals("message_test")) {  
    System.out.println("recebi a mensagem :"  
        + jObs.getString("message"));  
}
```

Figura 4.8 - Programação na aplicação em java que lida com a recepção de mensagens

A aplicação em java identificará que a mensagem que recebeu tem um parâmetro “type” igual a “message_test”, o que significa que a aplicação Android está a enviar uma mensagem, sendo exibida em seguida, tal como se verifica na figura 4.8.

URL	Content
.93. .228/loginUser.php	É uma organização? Login de um particular Name: <input type="text"/> Password: <input type="password"/> <input type="button" value="Login"/>
.93. .228/loginOrg.php	Login de uma organização Name: <input type="text"/> Password: <input type="password"/> <input type="button" value="Login"/>

Figura 4.9 - Páginas de *login*, particular e organização

```

if (jobs.getString("type").equals("login")) {
    BasicProperties replyProp = new BasicProperties().builder()
        .correlationId(properties.getCorrelationId()).build();

    String username = jobs.getString("username");
    String password = jobs.getString("password");
    List<User> users = dbClient.view("Users/UserNames")
        .includeDocs(true).key(username).query(User.class);
    List<LoginTime> userlogin = dbClient.view("Users/Logins")
        .includeDocs(true).key(username).query(LoginTime.class);
    JSONObject resp = new JSONObject();

    resp.put("type", "login_resp");
    if (users.size() == 0
        || !(users.get(0).getName().equals(username) && users
            .get(0).getPassword().equals(password))) {
        resp.put("error", "404");
        resp.put("description", "Fail login");
    } else {
        resp.put("error", "0");

        // Update login time.
        User currentUser = users.get(0);

        LoginTime lt = new LoginTime();
        lt.setUserId(currentUser.get_id());
        lt.setTime(new Date());

        dbClient.save(lt);
    }

    channel.basicPublish("", properties.getReplyTo(), replyProp, resp
        .toString().getBytes());
}

```

Figura 4.10 - Programação na aplicação em java que lida com um pedido de *login*

No caso das páginas web como é demonstrado na figura 4.9, o login é feito recorrendo ao *web service*. No entanto, na aplicação Android é feito um pedido à aplicação em java, com o percurso já referido na descrição da figura 4.7. Assim que a aplicação identifica que a mensagem vem com um parâmetro “type” igual a “login”, esta vai ver à base de dados recorrendo a “View” se existe algum utilizador com o nome recebido. Caso o nome e palavra passe recebida seja igual ao que se encontra na base de dados é criada uma resposta com “error” igual a zero. Caso contrário “error” igual a 404 com uma descrição de “Fail login”, tal como se verifica na figura 4.10.

The image shows two side-by-side browser windows. The left window, titled '.93. .228/menuUser.php', contains the following elements: a link 'Ver os sensores que lhe pertencem', a section separator '-----', the text 'Remova um utilizador(apagar-se)', a 'Nome:' label with an input field, a 'Remover utilizador' button, and another section separator. The right window, titled '.93. .228/menuOrg.php', contains: a link 'Veja os seus usernames', the text 'Adicione um utilizador', a 'Nome:' label with an input field, a 'Password:' label with an input field, an 'Adicionar utilizador' button, a section separator, the text 'Remova um utilizador', a 'Nome:' label with an input field, and a 'Remover utilizador' button.

Figura 4.11 - Página inicial dos utilizadores e das organizações

Caso o login, seja efectuado com sucesso, recorrendo para isso a uma página, figura 4.6, vai para as páginas presentes na figura 4.11.

The image shows a browser window titled '.93. .228/getUserOrg.php'. It features a link 'Veja os seus usernames', the text 'Adicione um utilizador', a 'Nome:' label with an input field containing 'joao', a 'Password:' label with an input field containing three dots, and an 'Adicionar utilizador' button. To the right of the form, the text 'joao pertence a FCT' and 'fctuser pertence a FCT' is displayed.

Figura 4.12 - Exemplo de adicionar um utilizador a uma organização

Inicialmente, foi feito o *login* de uma organização FCT, onde já tinha um utilizador fctuser. Após ser adicionado um utilizador chamada joao, imediatamente se pode verificar que foi associado correctamente à FCT, bastando para isso carregar em “Veja os seus *usernames*”, tal como se confirma na figura 4.12.

The image shows two side-by-side browser windows. The left window, titled '.93. .228/registerUser.php', contains the text 'Faça o registo do novo utilizador', a 'Nome:' label with an input field, a 'Password:' label with an input field, a 'Registar' button, and a link 'Se for uma organização carregue aqui'. The right window, titled '.93. .228/registerOrg.php', contains the text 'Faça o registo da sua organização', a 'Nome:' label with an input field, a 'Pass:' label with an input field, a 'Publica/Privado:' label with an input field, and a 'Registar' button.

Figura 4.13 - Página de registo de particulares e organizações

Nas páginas da figura 4.13 poderá registar-se, caso seja a primeira vez que vai utilizar o serviço. Podendo posteriormente assim utilizar essas credenciais, seja no *smartphone*, como nas páginas de login.

```
@WebMethod(operationName = "register_new_user")
public String register_new_user(@WebParam(name = "param1") String name, @WebParam(name = "param2") :
    if (name == null || password == null)
        return "Name and/or pass are null!";

    CouchDbClient dbClient = connectToDataBase();

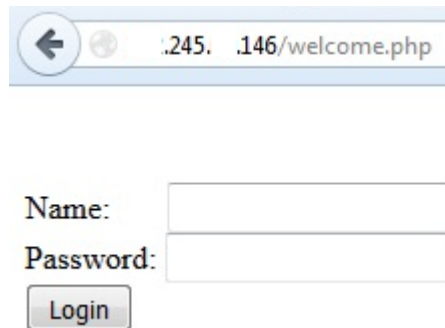
    List<User> users = dbClient.view("Users/UserNames").includeDocs(true)
        .key(name).query(User.class);

    User new_user = null;
    if (users.size() == 0) {
        new_user = new User();
        new_user.setName(name);
        new_user.setPassword(password);
        new_user.setActive(active);
        new_user.setType(type);

        Response resp = dbClient.save(new_user);
        new_user.set_id(resp.getId());
        new_user.set_rev(resp.getRev());
        return "registadocomsucesso";
    } else {
        return "ja existe User:" + users.get(0).getName() + " Pass: " + users.get(0).getPassword();
    }
}
```

Figura 4.14 - Serviço do *Web Service* register_new_user

Assim que se carrega no botão registar do lado esquerdo da figura 4.13, é chamado um serviço pertencente ao *web service*. Tendo em conta que é o registo de particulares, o serviço tem de nome register_new_user, tal como está na figura 4.14. Este serviço precisa dos seguintes parâmetros de entrada: nome, palavra passe, se está activo ou não e que tipo de utilizador é. Inicialmente cria-se uma ligação com a base de dados. Em seguida é verificado se já existe algum utilizador com o mesmo nome. Posteriormente é criado um utilizador new_user do tipo User, sendo que todos os campos desse utilizador são preenchidos com os dados recebidos. Caso ainda não exista nenhum utilizador com o nome escolhido, a informação é guardada na base de dados e retorna uma mensagem com “registadocomsucesso”. Caso contrário devolve “já existe User: +nome Pass: + palavra passe”.



← 192.168.1.245:8080/welcome.php

Name:

Password:

Login

Figura 4.15 - Página de login de um suposto cliente

Como se repara, na figura 4.15, o IP mudou, sendo outro servidor, denominado de Cliente.



Figura 4.16 - Página inicial do cliente

Após se fazer o login, em nome de fctuser e a sua palavra passe, o utilizador depara-se com a página, representada na figura 4.16. Ao fundo da página dá a indicação de qual é a organização a que pertence, caso fosse um particular estaria presente apenas o nome do utilizador. A opção de fazer LogOut, que consiste em desligar a sessão e apagar tudo o que esteja relacionado com a mesma, já se encontra disponível, bastando para isso carregar em Logtest.



Figura 4.17 - Página com informação de sensores

Após se carregar em sensores, tem-se acesso aos nomes e tipos de sensores que o utilizador tem associados à sua conta, tal como também a mensagem em Json, como está na figura 4.17.

```
public abstract java.lang.String fct.mgse.webservice.MGSEWS.addNewSensor(java.lang.String,java.lang.String,java.lang.String,java.lang.String)
addNewSensor ( 4321 , FCT , Temp , Org )
```

Figura 4.18 - Adicionar sensor a uma organização

O sensor teve de ser adicionado manualmente para teste, porque os sensores só podiam ser adicionados supostamente quando o cliente está a instalar o serviço, com o seu computador perto de si, de forma a configurá-lo. Os sensores foram adicionados directamente através do *web service* emulado a partir do netbeans, tal como se demonstra na figura 4.18.

Method returned

```
java.lang.String : "Sensor adicionado com sucesso"
```

Figura 4.19 - Mensagem de confirmação

Quando um sensor é adicionado com sucesso, aparecerá a mensagem presente na figura 4.19, que foi previamente definida nas funções do *web service*.

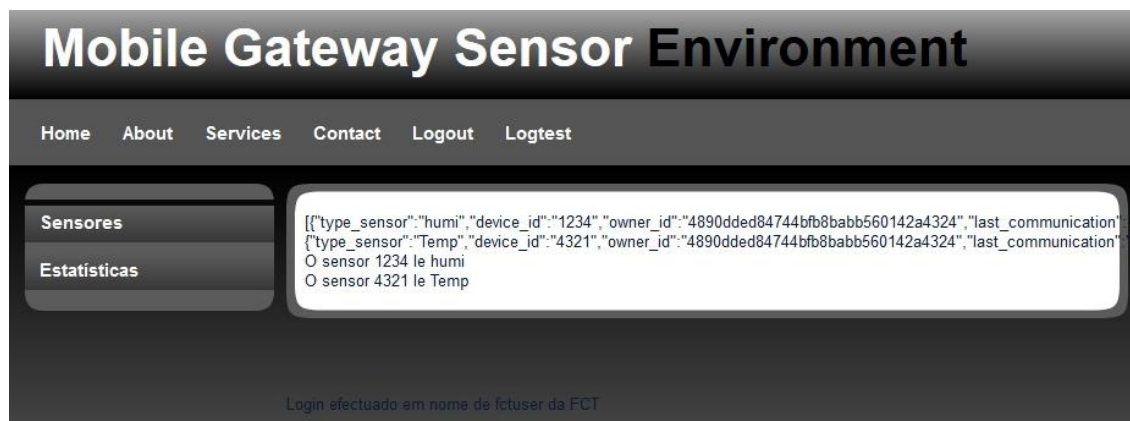


Figura 4.20 - Página com informação de sensores actualizado

Após se ter adicionado o novo sensor verifica-se que as mudanças foram efectuadas com sucesso, na figura 4.20.

O exemplo aqui demonstrado foi com um utilizador de uma organização, mas seria igual para um utilizador particular.

Praticamente em todos os botões onde se carrega está uma função do *web service* embutida.

Tal como foi indicado no capítulo *Web services*, é necessário um ficheiro WSDL no projecto, onde é possível verificar todas as funções que o utilizador tem à sua disposição, os outputs, os inputs e o endereço onde o *web service* está a correr. Este ficheiro encontra-se no anexo1.

```
$client = new SoapClient("http:// . . . :8080/MGSE_WS/MGSE_WS?wsdl",
    array("exceptions" => 0, "trace" => 1, 'stream_context' => stream_context_create(array(
        'http' => array(
            'header' => "Username: " . $user . "\r\nPassword: " . $finalhash))));
$result = $client->GetLogin();
```

Figura 4.21 - Exemplo usado no projecto que demonstra a forma como se cria um cliente SOAP

O SoapClient (configuração na figura 4.21) e o NuSoap são bibliotecas que se pode usar para poder aceder a *web services*, sendo que é necessário indicar qual o local na internet onde a aplicação de *web service* está a correr, para poder assim aceder às funções que se tem ao nosso dispor. Pode-se passar informação pelo *header* (cabeçalho em inglês), neste exemplo o nome de utilizador e a palavra passe, a fim de se saber se o utilizar pode realizar o *login*. A função retorna “foundtrue”, caso os dados sejam verdadeiros e “error”, caso seja falso.

Se se fizer \$client->getError(), caso tenha acontecido algum erro, ter-se-á acesso ao mesmo.

```

$param = array(
    'param1' => $name,
    'param2' => $finalhash,
    'param3' => $active,
    'param4' => $type
);
$result = $client->call('register_new_user', array('parameters' => $param), '', '', false, true);

```

Figura 4.22 - Exemplo de chamar uma função com parâmetros

Também é possível passar dados por parâmetro, tal como se pode verificar na figura acima. A estrutura é um pouco diferente da mostrada anteriormente, pois é utilizada a biblioteca NuSoap, tal como se verifica na figura 4.22.

Tanto é possível enviar parâmetros por cabeçalho, como por parâmetro nas funções.

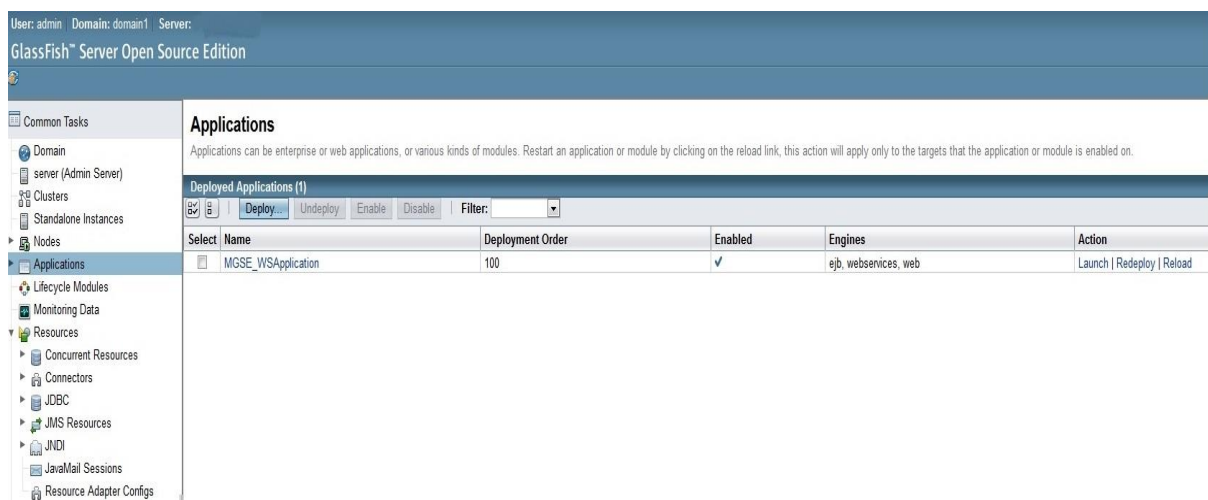


Figura 4.23 - Web service a correr no GlassFish

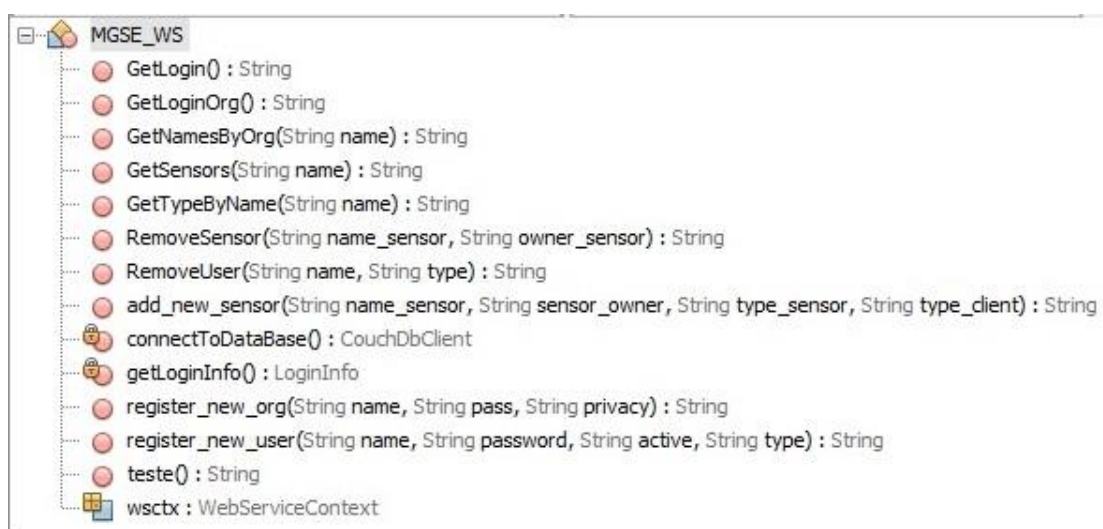



Figura 4.24 - Serviços criados no netbeans que estão disponíveis no web service

Após seguir o tutorial, acerca da criação de um *web service* no netbeans, é gerado um ficheiro com extensão war. O ficheiro .war (Web Application Resource) contem toda a informação acerca do *web service*, sendo necessário um programa que interprete este ficheiro. Recorreu-se ao GlassFish [76], sendo que precisa de estar instalado num servidor, este vai permitir criar um *web service*, fazendo com que as funções fiquem acessíveis na internet.

Na figura 4.24 consegue-se identificar todas as funções/ serviços, que estão disponíveis no *web service*. Na figura 4.23, verifica-se que está uma aplicação a correr, aplicação essa de nome MGSE_WSEApplication é o *web service* criado, através do ficheiro com extensão war. Desta forma estes serviços que foram criados para este projecto poderão ser acedidos em qualquer local, desde que se saiba o IP e a forma de interagir com ele.



```
private CouchDbClient connectToDataBase() {  
    // Connect with database.  
    CouchDbProperties properties = new CouchDbProperties()  
        .setDbName("mgse").setCreateDbIfNotExist(true)  
        .setProtocol("http").setHost(" . . . ") .setPort(5984)  
        .setUsername("mgse_reader").setPassword(" ")  
        .setMaxConnections(100).setConnectionTimeout(0);  
  
    return new CouchDbClient(properties);  
}
```

Figura 4.25 -Serviço disponibilizado no *web service* que é usado para se conectar à base de dados

Todas as funções implementadas no *web service* se tiverem de recorrer à base de dados, terão de chamar a função representada na figura 4.25, para assim criar uma variável do tipo CouchDBClient. Só assim se conseguirá aceder aos dados na base de dados.

```

@WebMethod(operationName = "GetLogin")
public String GetLogin() {
    MessageContext mctx = wsctx.getMessageContext();

    // Use the request headers to get the details
    Map http_headers
        = (Map) mctx.get(
            MessageContext.HTTP_REQUEST_HEADERS);
    List<String> userList = (List) http_headers.get("Username");
    List<String> passList = (List) http_headers.get("Password");

    if (userList == null || passList == null) {
        return null;
    }

    LoginInfo loginInfo = new LoginInfo(userList.get(0), passList.get(0));

    CouchDbClient dbClient = connectToDataBase();

    List<User> users = dbClient.view("Users/UserNames")
        .includeDocs(true)
        .key(loginInfo.getUsername())
        .query(User.class);
    String value = "null";
    //users.isEmpty() ||
    if (users.isEmpty())
        || !(users.get(0).getName().equals(loginInfo.getUsername())
            && users.get(0).getPassword().equals(loginInfo.getPassword().toString())) {
        return value = "error";
    }

    return value = "foundtrue";
}

```

Figura 4.26 -Função GetLogin do web service

Anteriormente na figura 4.21 foi mostrado como se colocava informação nos cabeçalhos da página php no servidor cliente, e na figura 4.26 verifica-se como se consegue ir buscar essa informação. Essa informação está nos HTTP_REQUEST_HEADERS e como se sabe de antemão que foram guardados com o nome de “Username” e “Password”, só é necessário guardar-se essa informação numa variável. Após saber-se qual é o utilizador que está a tentar fazer o login com determinada palavra passe, esses dados vão para a função LoginInfo, apenas para colocar mais atraente esteticamente, fazendo para isso uso de uma classe, que tem os atributos de um utilizador. No entanto, esta fase podia não ser feita e usava-se as variáveis directamente. Em seguida cria um cliente do tipo CouchDBClient chamado dbClient, capaz de aceder à base de dados. Antes de continuar é necessário explicar o conceito de view, é a principal ferramenta usada para fazer consulta dos documentos na couchDB. Há dois tipos de views, a permanente e a

temporária. As permanentes são guardadas numa secção chamada “design documents”, em contrapartida as temporárias não são guardadas na base de dados, apenas são executadas quando são chamadas.

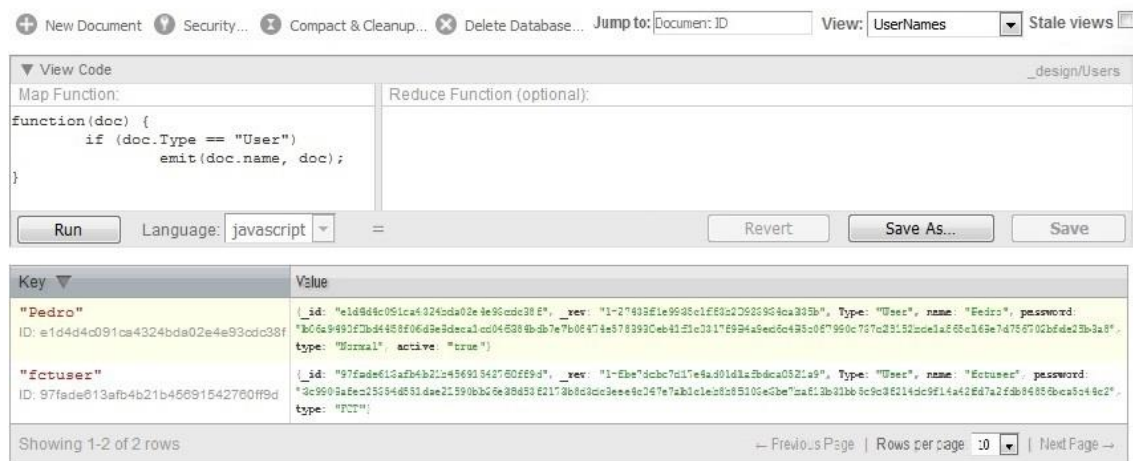


Figura 4.27 -Interface Futon do CouchDB na parte das *view* e respectivo código

Na figura 4.26, verifica-se que através do cliente dbClient vai-se aceder à *view*, feita na base de dados. Para tal é necessário dizer qual o grupo e qual o nome da *view*, “Users/Usernames”, com a chave do nome do utilizador a fazer o login, isto para filtrar os restantes utilizadores. Foi necessário também indicar qual a classe a que pertencia os dados que se ia receber. Caso o utilizador e palavra passe recebido, fosse igual ao que está na base de dados, devolve “foundtrue”, caso contrário “error”.

Analisando a interface Futon, representada na figura 4.27 no campo *view code*, pode-se verificar o código em javascript que é necessário, para devolver todos os ficheiros que tenham um campo Type igual a “User”. Sendo que no campo da “key” vai devolver o que está no campo “name” do documento e no campo “value” o documento todo. Confirma-se ainda que se está a visualizar a *view* que foi invocada na função GetLogin, tal como se pode verificar na figura 4.26. No campo “value” verifica-se ainda que a palavra passe encontra encriptada com SHA-512 para segurança do utilizador.



Figura 4.28 - Ficheiro exemplo do que é guardado no CouchDB

Na figura 4.28 verifica-se que os ficheiros na CouchDB estão em formato Json.

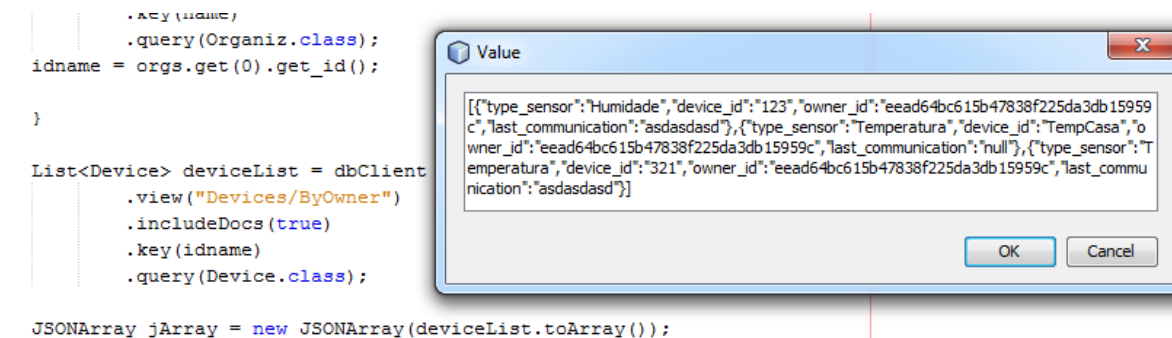


Figura 4.29 - Resposta do CouchDB a uma view

Além dos ficheiros serem guardados em Json, a resposta do CouchDB, também é dada em Json, tal como se pode comprovar na figura 4.29, o que facilita bastante a percepção dos dados. Neste caso invocou-se função `getSensors` com o parâmetro de entrada Pedro, o que significa que vai devolver os sensores que pertencem ao utilizador Pedro. O *web service* faz um pedido à base de dados através das *view's* que forem necessárias e esta devolve o seu resultado em Json.

Vai ser explicado em seguida os diferentes serviços existentes no *web service*.

`GetLogin()` : Recebe os dados a verificar através do cabeçalho e verifica se os dados do utilizador correspondem com os que estão na base de dados, enviando uma mensagem de confirmação.

`GetLoginOrg()` : Recebe os dados a verificar através do cabeçalho e verifica se os dados de uma organização correspondem com os que estão na base de dados, enviando uma mensagem de confirmação.

GetNamesByOrg(String name) : Através do nome da organização, devolve todos os utilizadores que lhe estão associados.

GetSensors(String name) : Mediante o nome de utilizador/organização, devolve os sensores que lhe estão associados.

GetTypeByName(String name) : Recebe um nome e verifica se esse nome pertence a uma organização ou é um particular, enviando uma mensagem de confirmação.

RemoveSensor(String name_sensor, String owner_sensor) : Recebe o nome do sensor e a quem lhe pertence, para posteriormente o apagar, enviando uma mensagem de confirmação.

RemoveUser(String name, String type) : Recebe o nome do utilizador e o seu tipo, para posteriormente o apagar, enviando uma mensagem de confirmação.

add_new_sensor(String name_sensor, String sensor_owner, String type_sensor, String type_client) : Recebe o nome do sensor, a quem lhe pertence, o tipo de sensor e o tipo de cliente, verificando se o utilizador já tem um sensor com esse nome, enviando uma mensagem de confirmação.

connectToDataBase() : Devolve uma variável do tipo CouchDbClient, capaz de se conectar à base de dados.

getLoginInfo() : Igual ao GetLogin(), usado para testes.

register_new_org(String name, String pass, String privacy) : Recebe o nome da organização, a palavra passe e se é privado ou público, enviando uma mensagem de confirmação.

register_new_user(String name, String password, String active, String type) : Recebe o nome de utilizador, a palavra passe, se está activo e o tipo de utilizador, enviando uma mensagem de confirmação.

Teste() : Devolve um conjunto de caracteres e é usado para testes.

Com estes dados e fornecendo o endereço IP do WSDL, seria possível a qualquer pessoa aceder ao *web service*.

4.4 Teste de performance

Foram feitos testes à base de dados com ferramentas disponibilizadas no site oficial que resultou nos seguintes resultados.

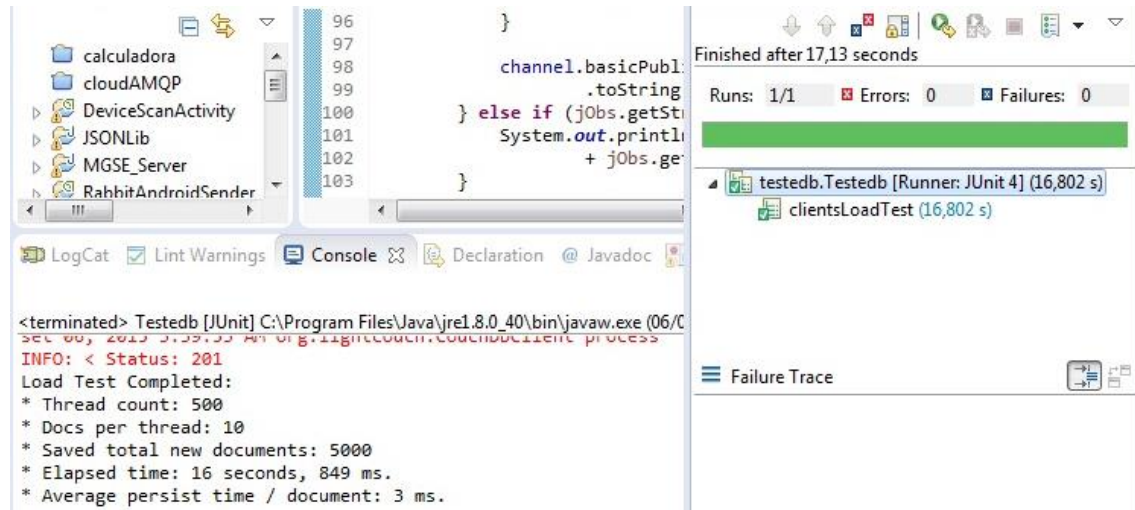


Figura 4.30 - Teste de performance feito à base de dados couchDB

Verificou-se que pela figura 4.30 que o desempenho era bastante satisfatório, pois dado as circunstâncias de usar um servidor grátis da Amazon, onde os recursos estão limitados. O facto de se ter conseguido guardar 5000 documentos no espaço de 16,8 segundos são resultados bastantes animadores.

Infelizmente não foi possível fazer testes aos outros componentes do projecto de forma automática. Foram feitos testes de forma manual, por exemplo, carregar 10 vezes no envio de mensagem e verificar que nenhuma das mensagens se perdeu.

5 Conclusão e Trabalhos Futuros

O objectivo da presente dissertação passava por criar uma forma que recolhesse a informação, sem que fosse necessário estar dependente de um ponto de internet e os sensores terem uma autonomia suficiente para não ser necessário uma fonte de alimentação externa. Posteriormente, seria possível visualizar a informação dos sensores na internet.

O objectivo e o projecto foram concretizados, mas no entanto é necessário salientar que o utilizador, apesar de poder instalar os sensores onde pretender, fica limitado por 3 factores:

- O número de pessoas que passa na zona com um telemóvel com ligação Bluetooth e a aplicação instalada
- Futura ligação à internet, pois se um utilizador recolher os dados, mas depois não tiver acesso à internet, o objectivo não é cumprido. Tal como se passar apenas uma pessoa por mês, e apesar de ficar com a informação dos sensores, esta poderá estar desactualizada
- Autonomia, tendo em conta a estimativa obtida na dissertação Sensores “em movimento”, o sensor principal onde está incluído o Arduino e o Bluetooth, durará meio ano. Já em comparação com os nós finais com os sensores acoplados, terão uma autonomia de 3,5 anos. Estes valores permitem com que se agende uma visita antecipadamente, assim que se verificar que a bateria está chegar ao tempo estabelecido

Houve alguns problemas iniciais na configuração do servidor da AMAZON, na tentativa de arranjar uma forma de aceder ao ambiente de trabalho para facilitar o desenvolvimento do projecto, mas acabou por se fazer através de SSH (Secure Shell) devido a tornar-se mais rápido.

Também houve alguns problemas com o GlassFish, por vezes, este deixava de funcionar sem razão aparente. Descobriu-se mais tarde que se tratava de um erro da versão instalada que tinha um *memory leak* (erro de memória). Isto levava a que toda a memória do servidor fosse consumida, por consequente este deixou de conseguir responder a qualquer pedido vindo do exterior. A solução passou por actualizar para uma nova versão, corrigindo assim o erro.

Para aceder a funções do *web service*, era necessário utilizar uma biblioteca. Começou-se por utilizar uma biblioteca já incluída no PHP, a SoapClient. No entanto, quando a informação era passada por cabeçalho funcionava sem problemas, mas quando se colocava parâmetros nas funções, este não retornava o que era suposto. Julgava-se ser um erro de programação, no entanto, tratava-se de um problema do SoapClient, este foi substituído pela NuSOAP sem que fosse detectado qualquer problema na sua utilização.

Houve alguns problemas com o material utilizado, nomeadamente o Arduíno/Bluetooth/leitor de cartões, que por vezes davam problemas e atrasaram o projecto.

O facto deste projecto ter sido realizado por duas pessoas diferentes, e por consequente, dividido em duas partes, foi bastante desafiante. Numa situação destas, é necessário que se estabeleça o que se pretende, pois em certos pontos o projecto unia-se e caso os dois intervenientes não estivessem na mesma linha de raciocínio, o conceito não funcionaria.

O facto da presente dissertação ter sido realizada em ambiente empresarial, neste caso na empresa IrRADIARE, foi uma mais valia para a percepção do seu ambiente e como lidar com a pressão existente.

O projecto apesar de concluído, há melhorias a fazer, salientando-se os seguintes pontos:

- Devido a usar-se o pacote gratuito da Amazon, não se tem acesso a todas as vantagens, nomeadamente o facto de ter apenas um servidor e não se usufrui da grande vantagem da *cloud*. Isto é, configurar-se 3 servidores, partindo do princípio que estão em diferentes locais, mesmo que 2 estejam em baixo, devido a uma casualidade, o servidor que resta continuará a funcionar sem qualquer problema.
- Implementar segurança SSL nas páginas que foram criadas em php, tanto as páginas no servidor AMAZON, como as páginas do cliente no servidor CLIENTE. Será necessário também encriptar os dados que são enviados do Arduíno para o *smartphone* e do *smartphone* para o Broker, recorrendo para isso a uma das tecnologias que foi mencionada no capítulo Segurança.
- Na aplicação feita em Android, colocar um número máximo de tentativas de login, tal como obrigar o sistema a desligar a sessão, caso esteja muito tempo activa.
- Recolha de informação de terceiros: foi algo que foi pensado, no entanto não foi implementado, mas tem tudo o que é necessário para se colocar essa ideia em prática. A ideia era criar um sistema de regalias a quem se disponibiliza-se a recolher informação de sensores que não lhes competia.
- Configurar o Broker de modo a guardar a informação no disco, caso o sistema onde o Broker está a funcionar se desligue. Actualmente, apenas retém a informação caso não haja um consumidor.
- Configurar o Tomcat de forma a alojar páginas da internet e criar um *web service*, recorrendo para isso à instalação do *plugin* Axis, libertando assim recursos, pois não seria necessário o GlassFish.
- Automatizar CouchDBLite, pois apesar de estar a funcionar e a guardar informação, tem de ser feito manualmente.
- A construção da base de dados do *smartphone* relativa às últimas actualizações, necessitará de ser revista, pois actualmente está pensada em guardar informação de 20 sensores. Mas o ideal

seria que essa base de dados fosse criada à medida que o utilizador encontrasse os sensores no seu percurso normal. O número máximo de informação dos diferentes sensores que o utilizador poderá querer guardar no seu *smartphone*, também poderia ser alterado, pois há *smartphones* que têm mais espaço e não lhes causará incómodo ao disponibilizar mais espaço para armazenar informação. Seria também vantajoso, fazer com que fosse possível efectuar um *reset* à base de dados do *smartphone*, contendo as últimas actualizações dos sensores, pois o utilizador pode mudar de zona e nunca mais passar pelos mesmos.

- Apesar de já se apresentar no site informação relativa aos sensores, ainda não se tem nenhum gráfico com medições efectuadas pelos sensores. Isto porque, o equipamento que era suposto ser utilizado inicialmente não chegou a tempo e teve de se fazer adaptações no projecto. Em vez de receber uma mensagem com o nome, com o tipo de sensor, a última comunicação e a quem pertence, seria enviar uma mensagem com a informação recolhida pelos sensores. Viria também em formato Json portanto, seria só necessário encontrar uma biblioteca que fosse capaz de desenhar gráficos em php e consequente actualização aos programas existentes.

6 Bibliografia

- [1] "IDC," [Online]. Available: <http://www.idc.com/prodserv/smartphone-os-market-share.jsp>. [Acedido em 06 06 2015].
- [2] AMAZON. [Online]. Available: <https://aws.amazon.com/pt/about-aws/whats-new/2006/08/24/announcing-amazon-elastic-compute-cloud-amazon-ec2---beta/>. [Acedido em 06 06 2015].
- [3] "Pplware," [Online]. Available: <http://pplware.sapo.pt/informacao/cloud-computing-o-futuro-da-computacao/>. [Acedido em 07 06 2015].
- [4] [Online]. Available: <http://techlozenge.com/images/Dropbox-Diagram.gif>. [Acedido em 07 06 2015].
- [5] [Online]. Available: <https://www.pinterest.com/pin/377598749977443118/>. [Acedido em 02 09 2015].
- [6] LEGO Lab, University of Aarhus, [Online]. Available: <http://legolab.daimi.au.dk/DigitalControl.dir/NXT/Actuators.html>. [Acedido em 17 06 2015].
- [7] "Faculdade de Engenharia da Universidade do Porto," [Online]. Available: <https://web.fe.up.pt/~eol/SP-leec/TRABALHOS/ROB/sensores.html>. [Acedido em 16 06 2015].
- [8] "EngineersGarage," [Online]. Available: <http://www.engineersgarage.com/articles/sensors>. [Acedido em 17 06 2015].
- [9] "Faculdade de Engenharia da Universidade do Porto," [Online]. Available: <https://web.fe.up.pt/~goii2000/M3/redes2.htm>. [Acedido em 14 06 2015].
- [10] [Online]. Available: <http://www.whatisnetworking.net/networking-topology/>. [Acedido em 02 09 2015].
- [11] "globalsign," [Online]. Available: <https://www.globalsign.com/pt-br/ssl-information-center/what-is-ssl.html>. [Acedido em 19 06 2015].
- [12] "globalsign," [Online]. Available: <https://www.globalsign.com/pt-br/ssl-information-center/types-of-ssl-certificate.html>. [Acedido em 19 06 2015].
- [13] "OVH," [Online]. Available: <https://www.ovh.pt/ssl/funcionamento-ssl.xml>. [Acedido em 19 06 2015].
- [14] J. E. Wilkes e V. K. Garg, Wireless and Personal Communications Systems, Prentice Hall, 1996.

- [15] P. Manteigas. [Online]. Available: <http://pedromanteigas.no.sapo.pt/encrypt.htm>. [Acedido em 19 06 2015].
- [16] M. Véstias, Redes Cisco para profissionais, FCA - Editora de Informática, 2009.
- [17] L. A. Barbosa, L. F. Braghetto, M. L. Brisqui e C. S. Silva, "Universidade Estadual de Campinas," 07 2003. [Online]. Available: <http://www.braghetto.eti.br/files/Trabalho%20Oficial%20Final%20RSA.pdf>. [Acedido em 02 09 2015].
- [18] R. R. Oliveira. [Online]. Available: <http://www.ronielton.eti.br/publicacoes/artigorevistasegurancadigital2012.pdf>. [Acedido em 02 09 2015].
- [19] D. Selent, "ADVANCED ENCRYPTION STANDARD," *RIVIER ACADEMIC JOURNAL*, vol. 6, 2010.
- [20] [Online]. Available: <http://keccak.noekeon.org/>.
- [21] "Link," [Online]. Available: http://www.link.pt/MicroSites/detalhe_artigo.aspx?idc=4846&idsc=5475&idl=1. [Acedido em 02 09 2015].
- [22] "Valarm," [Online]. Available: <http://www.valarm.net>. [Acedido em 02 09 2015].
- [23] [Online]. Available: <https://www.youtube.com/user/ValarmCorp>. [Acedido em 02 09 2015].
- [24] Libelium. [Online]. Available: <http://www.libelium.com/products/waspmote-mote-runner-6lowpan>. [Acedido em 2015 06 06].
- [25] "Libelium," [Online]. Available: <http://www.libelium.com/products/waspmote/wsn/>. [Acedido em 2015 06 06].
- [26] "SensorCloud," [Online]. Available: <http://www.sensorcloud.com>. [Acedido em 02 09 2015].
- [27] [Online]. Available: <http://es.slideshare.net/MelisaTudela/melisa-tudela-costa>. [Acedido em 02 09 2015].
- [28] J. Crispim. [Online]. Available: http://www.jose-crispim.pt/artigos/redes/redes_art/03_teorias.html. [Acedido em 04 09 2015].
- [29] W. Stallings, Wireless Communications & Networks Second Edition, Prentice Hall, 2005.
- [30] "Intel," [Online]. Available: http://www.intel.com/support/pt/wireless/wmax/5350_5150/sb/cs-028904.htm. [Acedido em 04 09 2015].
- [31] "Infowester," [Online]. Available: <http://www.infowester.com/wifi.php>. [Acedido em 04 09 2015].

- [32] "Cnet," [Online]. Available: <http://www.cnet.com/topics/networking/best-networking-devices/802-11ac>. [Acedido em 02 09 2015].
- [33] "NFC-forum," [Online]. Available: <http://nfc-forum.org/what-is-nfc/about-the-technology>. [Acedido em 02 09 2015].
- [34] "Infowester," [Online]. Available: <http://www.infowester.com/bluetooth.php>. [Acedido em 02 09 2015].
- [35] "Bluetooth," [Online]. Available: <http://www.bluetooth.com/Pages/History-of-Bluetooth.aspx>. [Acedido em 02 09 2015].
- [36] "Bluetooth," [Online]. Available: <http://www.bluetooth.com/Pages/Competing-Tech.aspx>. [Acedido em 02 09 2015].
- [37] [Online]. Available: <http://www.ictknowledgebase.org.uk/cloudcomputing101>. [Acedido em 02 09 2015].
- [38] "Infowester," [Online]. Available: www.infowester.com/cloudcomputing.php. [Acedido em 07 06 2015].
- [39] G. Magalhães. [Online]. Available: <http://protocoloti.blogspot.pt/2012/03/saas-paas-e-iaas-as-camadas-do-cloud.html>. [Acedido em 08 06 2015].
- [40] "Techtarget," [Online]. Available: <http://searchmobilecomputing.techtarget.com/definition/Advanced-Mobile-Phone-Service>. [Acedido em 17 07 2015].
- [41] A. S. Tanenbaum, Computer Networks Fourth Edition, Prentice Hall, 2002.
- [42] "Infowester," [Online]. Available: <http://www.infowester.com/2g.php>. [Acedido em 17 07 2015].
- [43] "SKYVIEW," [Online]. Available: http://skyview.pt/wa_files/gprs.pdf. [Acedido em 17 07 2015].
- [44] "Faculdade de Engenharia da Universidade do Porto," [Online]. Available: https://web.fe.up.pt/~amoura/comunicacoes/GSM_UMTS.pdf. [Acedido em 13 08 2015].
- [45] "Infowester," [Online]. Available: <http://www.infowester.com/3g4g.php>. [Acedido em 20 07 2015].
- [46] "Pplware," [Online]. Available: <http://pplware.sapo.pt/informacao/comparativo-ja-compensa-utilizar-4g/>. [Acedido em 13 08 2015].
- [47] "3gpp," [Online]. Available: <http://www.3gpp.org/technologies/keywords-acronyms/97-lte-advanced>. [Acedido em 11 08 2015].

- [48] "ZDNET," [Online]. Available: <http://www.zdnet.com/article/first-5g-mobile-networks-expected-in-2020-with-up-to-20-gbps-speeds/>. [Acedido em 11 08 2015].
- [49] "Computerworld," [Online]. Available: <http://www.computerworld.com/article/2931848/mobile-wireless/ready-or-not-and-its-not-5g-is-coming.html>. [Acedido em 11 08 2015].
- [50] J. F. Kurose e K. W. Ross, Computer networking, Pearson Education, Inc, 2012.
- [51] "mibqyyo," [Online]. Available: http://www.mibqyyo.com/pt-artigos/2014/12/29/evolucao-redes-4g-gsm-historia/#/vanilla/discussion/embed/?vanilla_discussion_id=0. [Acedido em 21 07 2015].
- [52] androidauthority, [Online]. Available: <http://www.androidauthority.com/sweet-spot-visual-look-delicious-history-android-563596/>. [Acedido em 27 03 2015].
- [53] [Online]. Available: <http://www.howtogeek.com/189036/android-is-based-on-linux-but-what-does-that-mean/>. [Acedido em 02 09 2015].
- [54] Google. [Online]. Available: <http://source.android.com/source/index.html>. [Acedido em 06 06 2015].
- [55] Google. [Online]. Available: <https://source.android.com/devices/>. [Acedido em 03 09 2015].
- [56] Google, [Online]. Available: <http://www.android.com/history/>. [Acedido em 06 06 2015].
- [57] Accunite Solutions, [Online]. Available: <http://www.techpluto.com/what-is-android/>. [Acedido em 06 06 2015].
- [58] [Online]. Available: <http://www.whatisios.org/>. [Acedido em 27 03 2015].
- [59] Vox Media, [Online]. Available: <http://www.theverge.com/2011/12/13/2612736/ios-history-iphone-ipad>. [Acedido em 27 Março 2015].
- [60] Apple. [Online]. Available: <https://developer.apple.com/library/ios/documentation/Miscellaneous/Conceptual/iPhoneOSTechOverview/iPhoneOSTechnologies/iPhoneOSTechnologies.html>. [Acedido em 03 09 2015].
- [61] S. L. Levin e S. Schmidt, "IPv4 to IPv6: Challenges, solutions, and lessons," 10 09 2014.
- [62] V. M. C. Leitão, "IPV6 - A NEW SECURITY CHALLENGE," UNIVERSIDADE DE LISBOA-Faculdade de Ciências, DI, 2011.
- [63] "Arsys," [Online]. Available: <http://www.arsys.pt/ajuda/directorio/infraestrutura-tecnica/ipv6.htm>. [Acedido em 14 07 2015].
- [64] A. Cruz, "Universidade do Porto," [Online]. Available:

- <http://civil.fe.up.pt/acruz/Mi99/asr/transicao.htm>. [Acedido em 14 07 2015].
- [65] “Infowester,” [Online]. Available: <http://www.infowester.com/dns.php>. [Acedido em 16 07 2015].
- [66] [Online]. Available: http://devedge.primedirective.net/viewsource/2002/soap-overview/index_pt_br.html. [Acedido em 02 09 2015].
- [67] J. C. Lima, “WEB SERVICES (SOAP X REST),” FACULDADE DE TECNOLOGIA DE SÃO PAULO, 2012 São Paulo. [Online].
- [68] “Infoq,” [Online]. Available: <http://www.infoq.com/br/articles/rest-soap-when-to-use-each>. [Acedido em 02 09 2015].
- [69] J. Black, “Ancient Origins,” [Online]. Available: <http://www.ancient-origins.net/ancient-writings-ancient-places-europe/dispilio-tablet-oldest-known-written-text-00913>. [Acedido em 2015 06 06].
- [70] R. Ramakrishnan e J. Gehrke, Database Management Systems 3ª edição, McGraw-Hill, 2003.
- [71] “Json,” [Online]. Available: <http://json.org>. [Acedido em 02 09 2015].
- [72] “Dataconomy,” [Online]. Available: <http://dataconomy.com/sql-vs-nosql-need-know/>. [Acedido em 04 09 2015].
- [73] “DigitalOcean,” [Online]. Available: <https://www.digitalocean.com/community/tutorials/understanding-sql-and-nosql-databases-and-different-database-models>. [Acedido em 04 09 2015].
- [74] “Pivotal,” [Online]. Available: <http://blog.pivotal.io/pivotal/case-studies/800000-messagesminute-how-nokias-here-uses-rabbitmq-to-make-real-time-traffic-maps>. [Acedido em 02 09 2015].
- [75] “Vmware,” [Online]. Available: <https://blogs.vmware.com/vfabric/2013/01/messaging-architecture-using-rabbitmq-at-the-worlds-8th-largest-retailer.html>. [Acedido em 02 09 2015].
- [76] “Glassfish,” [Online]. Available: <https://glassfish.java.net>. [Acedido em 02 09 2015].
- [77] [Online]. Available: <http://googlesystem.blogspot.pt/2007/11/google-launches-android-open-mobile.html>. [Acedido em 06 06 2015].
- [78] “Cloud Security Alliance,” [Online]. Available: <https://chapters.cloudsecurityalliance.org/brazil/2014/01/14/cloud-computing-tendencias-uma-nuvem-de-oportunidades/>. [Acedido em 09 06 2015].
- [79] [Online]. Available: <http://www.coisaetale.com.br/wp->

- content/uploads/2011/05/cloudcomputing.jpg. [Acedido em 07 06 2015].
- [80] “CCM,” [Online]. Available: <http://ccm.net/contents/145-cryptography-secure-sockets-layers-ssl>. [Acedido em 19 09 2015].
- [81] [Online]. Available: <http://www.palpitedigital.com/o-que-e-ssid-wireless/>. [Acedido em 22 08 2015].
- [82] “NIST,” [Online]. Available: http://csrc.nist.gov/groups/ST/hash/sha-3/winner_sha-3.html. [Acedido em 03 09 2015].
- [83] “Gsmarena,” [Online]. Available: <http://www.gsmarena.com/glossary.php3?term=hscsd>. [Acedido em 20 07 2015].
- [84] “Oracle,” [Online]. Available: <http://docs.oracle.com/javaee/6/tutorial/doc/gijvh.html>. [Acedido em 04 09 2015].
- [85] “Apache,” [Online]. Available: https://wiki.apache.org/couchdb/Introduction_to_CouchDB_views. [Acedido em 05 09 2015].
- [86] “RabbitMQ,” [Online]. Available: <https://www.rabbitmq.com/getstarted.html>. [Acedido em 02 09 2015].
- [87] “IrRADIARE,” [Online]. Available: <http://www.irradiare.com/>. [Acedido em 02 09 2015].

7 Anexo

Ficheiro WSDL usado no projecto, onde é possível verificar todas as funções que o utilizador tem à sua disposição, os outputs, os inputs e o endereço onde o *web service* está a correr.

```
<?xml version='1.0' encoding='UTF-8'?>
<!-- Published by JAX-WS RI at http://jax-ws.dev.java.net. RI's version is Metro/2.3 (tags/2.3-7528; 2013-04-29T19:34:10+0000) JAXWS-RJ/2.2.8 JAXWS/2.2 svn-revision#unknown. -->
<!-- Generated by JAX-WS RI at http://jax-ws.dev.java.net. RI's version is Metro/2.3 (tags/2.3-7528; 2013-04-29T19:34:10+0000) JAXWS-RJ/2.2.8 JAXWS/2.2 svn-revision#unknown. -->
<definitions name='WCSE_WS' targetNamespace='http://webservice.mgse.fct/' xmlns='http://schemas.xmlsoap.org/wsdl/' xmlns:xs='http://www.w3.org/2001/XMLSchema' xmlns:tns='http://webservice.mgse.fct/'
xmlns:soap='http://schemas.xmlsoap.org/wsdl/soap/' xmlns:wsa='http://www.w3.org/2007/05/addressing/metadata' xmlns:wsp_1_2='http://schemas.xmlsoap.org/ws/2004/09/policy' xmlns:wsp='http://www.w3.org/ns/ws-policy'
xmlns:wss='http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd'
  <types>
    <xs:schema>
      <xs:import schemaLocation='http://...:8080/WCSE_WS/WCSE_WS?xsd=1' namespace='http://webservice.mgse.fct/'/>
    </xs:schema>
  </types>
  <message name='GetSensors'>
    <part name='parameters' element='tns:GetSensors'/>
  </message>
  <message name='GetSensorsResponse'>
    <part name='parameters' element='tns:GetSensorsResponse'/>
  </message>
  <message name='GetNamesByOrg'>
    <part name='parameters' element='tns:GetNamesByOrg'/>
  </message>
  <message name='GetNamesByOrgResponse'>
    <part name='parameters' element='tns:GetNamesByOrgResponse'/>
  </message>
  <message name='teste'>
    <part name='parameters' element='tns:teste'/>
  </message>
  <message name='testeResponse'>
    <part name='parameters' element='tns:testeResponse'/>
  </message>
  <message name='register_new_user'>
    <part name='parameters' element='tns:register_new_user'/>
  </message>
  <message name='register_new_userResponse'>
    <part name='parameters' element='tns:register_new_userResponse'/>
  </message>
  <message name='register_new_org'>
    <part name='parameters' element='tns:register_new_org'/>
  </message>
  <message name='register_new_orgResponse'>
    <part name='parameters' element='tns:register_new_orgResponse'/>
  </message>
  <message name='GetLogin'>
    <part name='parameters' element='tns:GetLogin'/>
  </message>
  <message name='GetLoginResponse'>
    <part name='parameters' element='tns:GetLoginResponse'/>
  </message>
  <message name='GetLoginOrg'>
    <part name='parameters' element='tns:GetLoginOrg'/>
  </message>
  <message name='GetLoginOrgResponse'>
    <part name='parameters' element='tns:GetLoginOrgResponse'/>
  </message>
  <message name='RemoveUser'>
    <part name='parameters' element='tns:RemoveUser'/>
  </message>
  <message name='RemoveUserResponse'>
    <part name='parameters' element='tns:RemoveUserResponse'/>
  </message>
</definitions>
```

```

<message name="GetTypeByName">
  <part name="parameters" element="tns:GetTypeByName"/>
</message>
<message name="GetTypeByNameResponse">
  <part name="parameters" element="tns:GetTypeByNameResponse"/>
</message>
<message name="add_new_sensor">
  <part name="parameters" element="tns:add_new_sensor"/>
</message>
<message name="add_new_sensorResponse">
  <part name="parameters" element="tns:add_new_sensorResponse"/>
</message>
<message name="RemoveSensor">
  <part name="parameters" element="tns:RemoveSensor"/>
</message>
<message name="RemoveSensorResponse">
  <part name="parameters" element="tns:RemoveSensorResponse"/>
</message>
<portType name="MGSE_WS">
  <operation name="GetSensors">
    <input message="tns:GetSensors" wsam:Action="http://webservice.mgse.fct/MGSE_WS/GetSensorsRequest"/>
    <output message="tns:GetSensorsResponse" wsam:Action="http://webservice.mgse.fct/MGSE_WS/GetSensorsResponse"/>
  </operation>
  <operation name="GetNamesByOrg">
    <input message="tns:GetNamesByOrg" wsam:Action="http://webservice.mgse.fct/MGSE_WS/GetNamesByOrgRequest"/>
    <output message="tns:GetNamesByOrgResponse" wsam:Action="http://webservice.mgse.fct/MGSE_WS/GetNamesByOrgResponse"/>
  </operation>
  <operation name="teste">
    <input message="tns:teste" wsam:Action="http://webservice.mgse.fct/MGSE_WS/testeRequest"/>
    <output message="tns:testeResponse" wsam:Action="http://webservice.mgse.fct/MGSE_WS/testeResponse"/>
  </operation>
  <operation name="register_new_user">
    <input message="tns:register_new_user" wsam:Action="http://webservice.mgse.fct/MGSE_WS/register_new_userRequest"/>
    <output message="tns:register_new_userResponse" wsam:Action="http://webservice.mgse.fct/MGSE_WS/register_new_userResponse"/>
  </operation>
  <operation name="register_new_org">
    <input message="tns:register_new_org" wsam:Action="http://webservice.mgse.fct/MGSE_WS/register_new_orgRequest"/>
    <output message="tns:register_new_orgResponse" wsam:Action="http://webservice.mgse.fct/MGSE_WS/register_new_orgResponse"/>
  </operation>
  <operation name="GetLogin">
    <input message="tns:GetLogin" wsam:Action="http://webservice.mgse.fct/MGSE_WS/GetLoginRequest"/>
    <output message="tns:GetLoginResponse" wsam:Action="http://webservice.mgse.fct/MGSE_WS/GetLoginResponse"/>
  </operation>
  <operation name="GetLoginOrg">
    <input message="tns:GetLoginOrg" wsam:Action="http://webservice.mgse.fct/MGSE_WS/GetLoginOrgRequest"/>
    <output message="tns:GetLoginOrgResponse" wsam:Action="http://webservice.mgse.fct/MGSE_WS/GetLoginOrgResponse"/>
  </operation>
  <operation name="RemoveUser">
    <input message="tns:RemoveUser" wsam:Action="http://webservice.mgse.fct/MGSE_WS/RemoveUserRequest"/>
    <output message="tns:RemoveUserResponse" wsam:Action="http://webservice.mgse.fct/MGSE_WS/RemoveUserResponse"/>
  </operation>
  <operation name="GetTypeByName">
    <input message="tns:GetTypeByName" wsam:Action="http://webservice.mgse.fct/MGSE_WS/GetTypeByNameRequest"/>
    <output message="tns:GetTypeByNameResponse" wsam:Action="http://webservice.mgse.fct/MGSE_WS/GetTypeByNameResponse"/>
  </operation>
  <operation name="add_new_sensor">
    <input message="tns:add_new_sensor" wsam:Action="http://webservice.mgse.fct/MGSE_WS/add_new_sensorRequest"/>
    <output message="tns:add_new_sensorResponse" wsam:Action="http://webservice.mgse.fct/MGSE_WS/add_new_sensorResponse"/>
  </operation>
  <operation name="RemoveSensor">
    <input message="tns:RemoveSensor" wsam:Action="http://webservice.mgse.fct/MGSE_WS/RemoveSensorRequest"/>
  </operation>

```



```

        <input message="tns:RemoveSensor" wsam:Action="http://webservice.mgse.fct/MGSE_WS/RemoveSensorRequest"/>
        <output message="tns:RemoveSensorResponse" wsam:Action="http://webservice.mgse.fct/MGSE_WS/RemoveSensorResponse"/>
    </operation>
</portType>
- <binding name="MGSE_WSPortBinding" type="tns:MGSE_WS">
    <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>
    - <operation name="GetSensors">
        <soap:operation soapAction=""/>
        - <input>
            <soap:body use="literal"/>
        </input>
        - <output>
            <soap:body use="literal"/>
        </output>
    </operation>
    - <operation name="GetNamesByOrg">
        <soap:operation soapAction=""/>
        - <input>
            <soap:body use="literal"/>
        </input>
        - <output>
            <soap:body use="literal"/>
        </output>
    </operation>
    + <operation name="teste">
    - <operation name="register_new_user">
        <soap:operation soapAction=""/>
        - <input>
            <soap:body use="literal"/>
        </input>
        - <output>
            <soap:body use="literal"/>
        </output>
    </operation>
    - <operation name="register_new_org">
        <soap:operation soapAction=""/>
        - <input>
            <soap:body use="literal"/>
        </input>
        - <output>
            <soap:body use="literal"/>
        </output>
    </operation>
    - <operation name="GetLogin">
        <soap:operation soapAction=""/>
        - <input>
            <soap:body use="literal"/>
        </input>
        - <output>
            <soap:body use="literal"/>
        </output>
    </operation>
    - <operation name="GetLoginOrg">
        <soap:operation soapAction=""/>
        - <input>
            <soap:body use="literal"/>
        </input>
        - <output>
            <soap:body use="literal"/>
        </output>
    </operation>

```

```

- <operation name="RemoveUser">
  <soap:operation soapAction=""/>
  - <input>
    <soap:body use="literal"/>
  </input>
  - <output>
    <soap:body use="literal"/>
  </output>
</operation>
- <operation name="GetTypeByName">
  <soap:operation soapAction=""/>
  - <input>
    <soap:body use="literal"/>
  </input>
  - <output>
    <soap:body use="literal"/>
  </output>
</operation>
- <operation name="add_new_sensor">
  <soap:operation soapAction=""/>
  - <input>
    <soap:body use="literal"/>
  </input>
  - <output>
    <soap:body use="literal"/>
  </output>
</operation>
- <operation name="RemoveSensor">
  <soap:operation soapAction=""/>
  - <input>
    <soap:body use="literal"/>
  </input>
  - <output>
    <soap:body use="literal"/>
  </output>
</operation>
</binding>
- <service name="MGSE_WS">
  - <port name="MGSE_WSPort" binding="tns:MGSE_WSPortBinding">
    <soap:address location="http://...:8080/MGSE_WS/MGSE_WS"/>
  </port>
</service>
</definitions>

```

Anexo 1. : Ficheiro WSDL usado no projecto